UNIVERSIDADE FEDERAL DE PERNAMBUCO
DEPARTAMENTO DE ELETRÔNICA E SISTEMAS
MESTRADO PROFISSIONALIZANTE EM ENGENHARIA ELÉTRICA

Modelagem do SADI (Sistema de Acompanhamento a Doação de sangue no Interior do Estado do Amazonas) apoiada pela UML

por

Jorlene de Souza Marques
jorlene@hemoam.org.br

Orientação
Profa. Dra. Fernanda Maria Ribeiro de Alencar
fmra@ufpe.br

RECIFE – PE
JULHO/2003

JORLENE DE SOUZA MARQUES

Modelagem do SADI (Sistema de Acompanhamento a Doação de sangue no Interior do Estado do Amazonas) apoiada pela UML

Dissertação do Curso de Mestrado Profissionalizante em Engenharia Elétrica apresentada a Universidade Federal de Pernambuco - UFPE para obtenção do grau de Mestre, orientada pela Profa. Dra. Fernanda Maria Ribeiro de Alencar.

Recife - PE
Julho/2003

Jorlene de Souza Marques

Modelagem do SADI (Sistema de Acompanhamento a Doação de sangue no Interior do Estado do Amazonas) Apoiada pela UML

Dissertação apresentada como requisito parcial à obtenção do grau de mestre no Programa de Pós-Graduação em Engenharia Elétrica do Centro de Tecnologia e Geociências/Escola de Engenharia da Universidade Federal de Pernambuco

Aprovado em 25 de agosto de 2003.

BANCA EXAMINADORA

____

Fernanda Maria Ribeiro de Alencar
Orientadora

____

Rafael Dueire Lins

____

Jaelson Freire Brelaz de Castro

DEDICATÓRIA

A minha mãe (Orlene Braga Cabral de Sousa), pelo amor, carinho e estímulo que me ofereceu. Dedico-lhe essa conquista como gratidão.

AGRADECIMENTOS

Gostaria de agradecer a todos que contribuíram direta ou indiretamente para a realização deste. Infelizmente seria difícil citar os nomes de todos. No entanto, gostaria de nomear alguns:

- A Deus, autor, razão da minha fé e certeza de minha vitória.
- Aos meus pais, pelas angústias e preocupações que passaram por minha causa. A eles a quem devo minha vida, a certeza de meu amor.
- Às minhas irmãs Josie e Zila, pelos incentivos constantes.
- À minha avó, pelo carinho e força sempre ofertados.
- Ao meu marido Max Nunes, pelo companheirismo, paciência e ajuda no decorrer deste.
- Às professoras Rosiane Rodrigues e Tayana Conte, pelo apoio e contribuições a mim oferecidas; e Fernanda Alencar pelos conhecimentos, correções, pela dedicação, paciência, e sugestões na elaboração deste.
- Aos colegas de trabalho, em especial: Maykel Célio pelas constantes ajudas e incentivos, ainda, Luiza Yanai, Rogério Carminé, Eduardo Nascimento e Jean Sarto.
- Às colegas Laura Jane, Áurea Hiléia e Neide Alves pelos momentos de angústias e sofrimentos passados juntas.
- Aos amigos Marco Antonio, Claudia e Jeisa pela acolhida calorosa em Recife.
- À Direção do HEMOAM pelo apoio dado ao longo da realização do Mestrado.
- À todos os funcionários da Fundação HEMOAM pelas entrevistas cedidas.

SUMÁRIO

LISTA DE ABREVIATURAS

CMM	Capability Maturity Model
SEI	Software Engineering Institute
SPICE	Software Process Improvement and Capability dEtermination
UML	Unified Modeling Language
CASE	Computer-Aided Software Engineering
DSSD	Data Structured Systems Development
HEMOAM	Fundação de Hematologia e Hemoterapia do Amazonas
JSD	Desenvolvimento Estruturado de Jackson
SSD	System Specification Diagram
OMT	Object Modeling Technique
OOSE	Object-Oriented Software Engineering
OMG	Object Management Group
DBMS	Data Base Management System
OOA	Análise Orientada a Objeto
OOD	Projeto Orientado a Objeto
Web	World Wide Web
PSP	Personal Software Process
TSP	Team Software Process
PROSE	Processo Orientado a Objetos para Software Extensível
RUP	Rational Unified Process
HP	Hewlett Packard
PS	Postscript
EPS	Encapsulated Postscript
SVG	Scalable Vector Graphics
WMF	Windows Meta File
GIF	Graphical Interchance Format
PNG	Portable Network Graphics
JPG	Joint Photographic Group
VB	Visual Basic
SSADM	Strutured Systems Analysis and Design Methodology
HTML	Hypertext Markup Language
SQL	Structured Query Language
LAN	Local Área Networks
MAN	Metropolitan Area Networks
WAN	Wide Área Networks
ITU	International Telecommunications Union
BPS	Bits por segundo
MBPS	Megabits por segundo
KBPS	Kilobits por segundo
GBPS	Gigabits por segundo
ISDN	Integrated Services Digital Network

DSL	Digital Subscriber Line
ATM	Asynchronous Transfer Mode
SONET	Synchronous Optical Network
ISPs	Provedores de Serviços da Internet
SMDS	Switched Multimegabit Data Service
TCP/IP	Transmission Control Protocol / Internet Protocol
UCTs	Unidades de Coleta e Transfusão
SAD	Sistema Acompanhamento de Doações de sangue
QBC	Quantitative Buffy Coat
ABO	Grupos Sanguíneos A, B, O, AB
RAM	Random Acess Memory
SCSI	Small Computer System Interface
RAID	Redundant Array of Independent (or Inexpensive) Disks
DAT	Digital Áudio Tape
TXT	Formato Text
SADI	Sistema Acompanhamento de Doações de sangue do Interior
LP	Linha Privada
VPN	Virtual Private Network

LISTA DE FIGURAS

LISTA DE TABELAS

RESUMO

Este trabalho reúne elementos de estudo sobre metodologias e tecnologias para o desenvolvimento de aplicações, com a finalidade de utilizá-las no desenvolvimento de um sistema para controlar as doações de sangue nas Unidades de Coleta e Transfusão (UCTs) da Fundação HEMOAM. Unidades essas situadas no interior do Estado do Amazonas. Contribuindo, desta forma, com a comunidade universitária, com a administração da Fundação HEMOAM e com os doadores de sangue e pacientes do interior do Estado. Nesse estudo foi adotado um processo de desenvolvimento de software próprio, orientado a objeto, tendo por base o Processo Unificado da Rational – RUP. Assim, para a modelagem das UCTs foi considerada a Linguagem Modelagem Unificada da OMG - UML (*Unified Modeling Language*). A modelagem da aplicação permitiu entender a informação, a função, o comportamento e a completude do sistema, tornando a tarefa de análise de requisitos mais fácil e mais sistemática. A modelagem serviu de base para o projeto fornecendo uma representação essencial do software que no futuro será mapeada para o contexto da implementação.

ABSTRACT

This end-of-term paper gathers elements about methodologies and technologies of the applications, with the purpose to use them in development f a system to control donations of blood in the units of collection and transfusion (UCTs) of the Foundation Hemoam. Units situated in the interior of the state of Amazon. Contributing, this way, with the university community, with the administration of the Foundation Hemoam, e with the blood donors and patients of the interior of the State. In this study a own process of software development was adopted, object-oriented, based in the Rational Unified Process - RUP. Thus, for the modeling of the UCTs the Language of Modeling Unified of OMG – UML (*Unified Modeling Language*) was considered. The modeling of the application in allowed to understand them the information, the function, the behavior and totality of the system, becoming the task of more easy and more systematic analysis of requirements. The modeling served of base for the project supplying an essential representation of the software that in the future will be mapeada for the implementation context.

CAPÍTULO 1 – INTRODUÇÃO

Atualmente, uma grande parte da população mundial depende de sistemas de software para realizar suas atividades diárias. Software faz parte de nossas vidas e, embora muito já tenha sido conseguido na busca da qualidade e produtividade no desenvolvimento e manutenção de software nos últimos 30 anos, muito resta para ser feito.

A qualidade de produtos de software, entretanto, está fortemente relacionada à qualidade do processo de software. Assim, na década de 90, no Brasil houve um aumento da preocupação com a modelagem e melhorias no processo de software.

Abordagens importantes sugerem que, melhorando o processo de desenvolvimento de software, poderemos melhorar a qualidade dos produtos resultantes. São elas:

i. norma ISO 9000 (Rocha; Maldonado; Weber, 2001) voltadas para processos organizacionais em geral;

ii. norma ISO/ IEC 12207 (Rocha; Maldonado; Weber, 2001) voltada para processos de ciclo de vida de software;

iii. modelo CMM (*Capability Maturity Model*) do SEI (Software Engineering Institute) (Bartié, 2002) voltado para o processo de software na proposta de melhoria contínua, mais eficiente e controlado;

iv. modelo SPICE (*Software Process Improvement and Capability dEtermination*) (Rocha; Maldonado; Weber, 2001) também voltado para o processo de software.

A pesquisa em processo de software trata dos métodos e técnicas utilizados para avaliar, apoiar e melhorar as atividades de desenvolvimento e manutenção de software. A primeira

contribuição importante da pesquisa na área de processo de software é o convencimento de que desenvolver software é um esforço coletivo, complexo e criativo e de que a qualidade do software depende das pessoas, da organização e dos procedimentos usados em seu desenvolvimento (Rocha et al., 2001).

Rumbaugh, Booch, e Jacobson (2000), afirmam: “A modelagem é a parte central de todas as atividades que levam à implantação de um bom software”. É através dela que representamos de forma abstrata e simplificada um sistema real. Os modelos são úteis para:

i. compreender os problemas;
ii. comunicar decisões com os envolvidos no projeto;
iii. favorecer o processo de verificação e validação;
iv. capturar aspectos de relacionamentos entre os objetos observados;
v. servir como referencial para geração de estruturas de dados;
vi. estabelecer conceitos únicos a partir de visões diversas.

1.1.Motivação

A modelagem é uma técnica de engenharia aprovada e bem-aceita. Construímos modelos para compreender melhor o sistema que estamos desenvolvendo. Buscamos o entendimento do SADI como um todo.

A UML (Unified Modeling Language) (Rumbaugh; Booch; Jacobson, 2000) por sua vez, é uma linguagem de modelagem de sistemas bastante madura e conhecida tanto no meio profissional quanto acadêmico, que utiliza o paradigma orientado a objetos, permitindo a uniformização dos modelos usados para análise, projeto e implementação. Muitas empresas que não possuíam metodologia e que ainda estão galgando degraus na área de desenvolvimento para web, Estão adotando a análise orientada a objetos. Muitas empresas que utilizam o paradigma estruturado e que estavam buscando evolução e melhoria no

processo de desenvolvimento, também migraram para o paradigma orientado a objetos. Resta migrar, então, as empresas que estão satisfeitas com a análise estruturada, apesar de saberem que está faltando “algo mais” nos seus produtos, mas têm receio de investir e arriscar na mudança de paradigma. Quanto mais a análise orientada a objetos evolui, mais estas empresas estarão se aproximando da mudança.

Modelar, com o apoio da UML, um sistema para controlar as doações de sangue nas Unidades de Coleta e Transfusão da Fundação HEMOAM, possibilitará entender a informação, a função, e o comportamento de sistema como um todo; tornando a tarefa de análise de requisitos mais completa. Os requisitos serão documentados, especificados e modelados, modificando assim o processo de desenvolvimentos de sistemas da Fundação Hemoam (capital), que é considerado imaturo, pois depende fortemente dos profissionais que lá se encontram, quase não possui documentação formal, e de difícil controle gerencial.

A política atual da Fundação HEMOAM, no que diz respeito a informática, visa o aumento de produtividade de trabalho; a diminuição do custo de desenvolvimento e manutenção dos sistemas informatizados; a maior portabilidade e reutilização de código e finalmente a satisfação dos nossos usuários internos e externos. Acreditamos que: classes bem escritas hoje reduzem tempo e custo de desenvolvimento amanhã; investir tempo em desenvolver e evoluir, não em consertar falhas; e ter manutenções evolutivas e não corretivas.

A automação do acompanhamento da doação de sangue no interior do Estado do Amazonas. Que atualmente todo o controle é manual. As informações são anotadas em livros. Os resultados de exames sorológicos (realizados na capital) são enviados por fax, para as Unidades de Coleta e Transfusão (no interior do Estado do Amazonas), com a finalidade de liberar as bolsas de sangue para transfusão. Esses resultados de exames demoram, em alguns locais, até dois meses para chegar na mão do doador de sangue, pois são enviados pelo correio.

1.2.Objetivo

O objetivo principal é propor para o HEMOAM a modelagem de um sistema de informação que controle a doação de sangue no interior do Estado do Amazonas, com o auxílio da UML para:

i. adquirir conhecimentos sobre as metodologias, processos, ferramentas e tecnologias de comunicação de dados para o desenvolvimento de sistemas de informação;
ii. elaborar e disponibilizar esta dissertação como fonte de pesquisa para futuros projetos;

1.3.Sub-objetivos

Através do objetivo principal uma série de sub-objetivos podem ser alcançados, tais como:

i. adquirir conhecimentos para mudar o processo de desenvolvimento de software na Fundação HEMOAM, que atualmente é um processo improvisado pelos profissionais e gerentes;
ii. adotar uma metodologia de desenvolvimento de software para que as atividades, realizadas na Gerência de Sistemas do Hemoam, sejam controladas e documentadas. Contribuindo desta forma, com a administração da Fundação HEMOAM;
iii. eliminar o controle manual das informações sobre doação de sangue no interior do estado do Amazonas, quando a modelagem for implementada;

iv. propor um meio para troca de informações entre as UCTs e a sede do HEMOAM na capital Manaus.

1.4.Metodologia

Dessa forma, para alcançar os objetivos deste trabalho, é necessário fazer uma revisão sobre as várias técnicas para modelagem de sistemas de software; fazer uma revisão sobre alguns processos de desenvolvimentos de software; fazer levantamento sobre algumas tecnologias para comunicação de dados; fazer levantamento das necessidades do sistema SADI; fazer estudo de viabilidade geral justificando o desenvolvimento do sistema SADI; escolher um processo para o desenvolvimento do sistema SADI, fazendo parte do escopo deste trabalho, sua modelagem nas etapas de análise e projeto;

1.5.Descrição dos vários capítulos

O capítulo 2 desta dissertação aborda as técnicas para modelagem de software, contextualizando o leitor com um breve relato. O capítulo 3 focaliza a UML que é a linguagem de modelagem utilizada na modelagem do sistema do estudo de caso deste trabalho. O capítulo 4 abrange alguns processos de desenvolvimento de software encontrados na literatura e algumas ferramentas utilizadas no auxílio ao desenvolvimento. O capítulo 5 apresenta as principais tecnologias para comunicação de dados. O capítulo 6 contém a proposição de uma modelagem de um sistema de informação para controle da doação de sangue no interior do estado do Amazonas. Finalmente, o capítulo 7 apresenta os resultados obtidos no decorrer do trabalho, abordando dificuldades encontradas e sugestões para futuros trabalhos.

CAPÍTULO 2 – TÉCNICAS PARA MODELAGEM DE SISTEMAS DE INFORMAÇÃO

No passado a modelagem de sistemas de informação era realizada sem métodos ou formalismos, expressa apenas por textos em linguagem natural. A derivação da modelagem da fase de análise para a fase de projeto era feita sem nenhum critério. Os primeiros trabalhos de modelagem da análise iniciaram-se no final da década de 1960 e começo da década de 1970. No final da década de 1970 a técnica estruturada foi popularizada.

Ainda na década de 70, métodos orientados a objetos também começam a surgir, e as pessoas envolvidas com metodologias, que estavam diante de novas linguagens de programação orientadas a objetos e de aplicações cada vez mais complexas, começam a experimentar este novo método alternativo de análise e projeto. Quando a década de 1980 chegava ao fim, a técnica orientada a objeto começava a amadurecer.

Neste capítulo vamos abordar estas duas técnicas: técnica estruturada e a técnica orientada a objeto, além disso, vamos complementar a técnica estruturada com outra técnica alternativa de análise orientada a estrutura de dados, o método de Jackson (1983). Nosso objetivo não é comparar estas técnicas, mas sim conhecê-las.

2.1. A Técnica Estruturada

A Técnica Estruturada foi introduzida por uma série de trabalhos que foram realizados. Os pesquisadores Stevens, Myers e Constantine (1974) e Yourdon e Constantine (1978) sentiram necessidade de uma notação gráfica que representasse os dados e os processos que

os transformavam. Posteriormente estes processos foram mapeados numa arquitetura de projeto.

A Técnica Estruturada foi popularizada por DeMarco (1979). Ele introduziu e nomeou símbolos gráficos que possibilitaram ao analista criar modelos do fluxo da informação. Ele sugeriu o uso de um *dicionário de dados* e de *narrativas de processamento* para complementar os modelos do fluxo da informação.

Nos anos que se seguiram, variações da Técnica Estruturada foram sugeridas por Page-Jones (1980), Gane e Sarson (1982) entre outros. Extensões de tempo real foram introduzidas por Ward e Mellor (1985) e posteriormente por Hatley e Pirbhai (1987). A Técnica Estruturada passa a ser apoiada por uma série de ferramentas CASE (Computer-Aided Software Engineering) que auxiliam a criação de cada elemento do modelo, Yourdon (1992).

2.1.1.Uma visão da Técnica Estruturada

A Técnica Estruturada começa com um único processo ou função que representa o propósito geral do software a ser desenvolvido. Os processos complexos são divididos recursivamente, até que reste um com muitas pequenas funções fáceis de serem implementadas.

Na decomposição funcional um sistema é encarado basicamente como o fornecedor de uma ou mais funções para o usuário final. A decomposição de um processo em sub-processos é um tanto arbitrária. Pessoas diferentes produzirão decomposições diferentes.

Na Técnica Estruturada há uma dificuldade em garantir compatibilidade entre as fases de análise e projeto e, posteriormente, do projeto para a implementação. Na maioria das vezes, grandes alterações ou extensões dos modelos criados geram um grande esforço. Não podemos esquecer que a comunicação entre desenvolvedores e usuários é difícil, em virtude

dos diagramas não serem muito expressivos fora da compreensão da equipe de desenvolvimento.

Segundo Rumbaugh, Blaha, Premerlani, Eddy e Lorensen (1994):

> Um projeto onde se utiliza a Técnica Estruturada tem os limites do sistema claramente definidos, através dos quais os procedimentos do software devem comunicar-se com o mundo real. A estrutura de um projeto segundo a Técnica Estruturada é derivada, em parte, do limite do sistema, o que pode tornar difícil ampliar um projeto até um novo limite.

Uma outra dificuldade desta abordagem é o tratamento do banco de dados. É difícil mesclar o código de programação organizado por funções com um banco de dados organizado em torno dos dados.

Na Técnica Estruturada são abordadas diversas notações para a especificação formal do software. Durante a fase de análise, são utilizados:

i. diagramas de fluxo de dados;

ii. especificações de processos;

iii. dicionário de dados;

iv. diagramas de transições de estados;

v. diagramas de entidades-relacionamentos.

Na fase de projeto, são acrescentados detalhes aos modelos de análise. Nessa fase ainda, os diagramas de fluxo de dados são convertidos em descrições de *diagramas de estruturas* representando a codificação em linguagem de programação.

Segundo Yourdon (1992), "o diagrama de fluxo de dados ilustra as funções que o sistema deve executar; e consiste em processos, depósitos de dados, fluxos e terminais". Ele afirma também que "embora o diagrama de fluxo de dados ofereça uma prática visão geral dos principais componentes funcionais do sistema, não fornece qualquer detalhe sobre esses componentes". Para mostrar os detalhes de *qual* informação é transformada e como é

transformada existe a necessidade de duas ferramentas de suporte textual de modelagem: o *dicionário de dados* e a *especificação de processos*.

Conforme Rumbaugh et. al (1994) :

> Os diagramas de fluxo de dados modelam as transformações dos dados à medida que fluem através do sistema e são o ponto central da Técnica Estruturada. Um diagrama de fluxo de dados consiste em processos, fluxos de dados, atores e depósitos de dados. Começando do diagrama de fluxo de dados de mais alto nível, a Técnica Estruturada divide recursivamente processos complexos em diagramas de nível mais baixo, até que haja muitos processos de implementação simples. Quando os processos resultantes estão suficientemente simples, a decomposição é interrompida e é escrita uma especificação para cada processo do nível mais baixo. As especificações de processos podem ser expressas através de tabelas de decisão, pseudocódigo ou outras técnicas. O dicionário de dados contém os detalhes que faltam nos diagramas de fluxo de dados. Ele define os fluxos de dados e depósitos de dados e o significado de vários nomes.

Yourdon (1992) ressalta que “o diagrama de transições de estado focaliza o comportamento tempo-dependente do sistema, a seqüência na qual se tem acesso aos dados e em que as funções serão executadas”.

Segundo Rumbaugh et. al (1994) “os diagramas de transições de estados modelam o comportamento tempo-dependente e são semelhantes ao modelo dinâmico. A maioria dos diagramas de transições de estados descreve os processos ou o controle do tempo da execução de funções e dos acessos aos dados disparados por eventos”.

Conforme Yourdon (1992), “os diagramas de entidades-relacionamentos dão ênfase aos relacionamentos de dados”. O autor afirma também que “o diagrama de entidades-relacionamentos possui dois importantes componentes: tipos de objetos e relacionamentos”.

Rumbaugh et. al (1994) afirma que “os diagramas de entidades-relacionamentos ressaltam os relacionamentos entre os depósitos de dados que, de outra forma, somente seriam vistos na especificação de processos. Cada elemento de dado do ER corresponde a um depósito de dados do diagrama de fluxo de dados”.

Finalizando, os autores Rumbaugh et. al (1994) ressalta que:

> O projeto estruturado segue-se à análise estruturada e aborda os detalhes de nível inferior. Por exemplo, durante o processo estruturado, os processos do diagrama de fluxo de dados são agrupados em tarefas e alocados aos processos do sistema operacional e às CPUs. Os processos do diagrama de fluxo de dados são convertidos em funções da linguagem de programação, e é criado um diagrama de estruturas que mostra a árvore de chamadas de procedimentos.

Nas seções que seguem, conforme já citamos, serão apresentadas as técnicas de *análise orientada a estrutura de dados* e *análise orientada a objetos*.

2.2.Técnicas Alternativa de Análise Orientada a Estrutura de Dados

As Técnicas Alternativas de Análise Orientada a Estrutura de Dados possuem diversas características idênticas a Técnica Estruturada apresentada anteriormente, mas cada uma delas têm aspectos que as tornam únicas.

É importante ressaltar que o domínio da informação de um problema de software abrange o fluxo, conteúdo e estrutura. As Técnicas Alternativas de Análise Orientada a Estrutura de Dados representam os requisitos de software ao se concentrarem na estrutura de dados e não no fluxo de dados.

Dentre as Técnicas Alternativas de Análise Orientada a Estrutura de Dados enumeraremos as que seguem:

1. o diagrama de Warnier (Warnier ,1974);
2. a abordagem DSSD (Data Structured Systems Development) de Ken Orr (Orr,1977);
3. o Desenvolvimento Estruturado de Jackson (Jackson ,1983).

Para ilustrar melhor o que foi dito sobre as Técnicas Alternativas de Análise Orientada a Estrutura de Dados, vamos descrever sucintamente apenas o Desenvolvimento Estruturado de Jackson.

2.2.1.Desenvolvimento Estruturado de Jackson (JSD)

O Desenvolvimento Estruturado de Jackson (JSD) surgiu do trabalho realizado por Michael A. Jackson (1975) (1983) sobre a análise do domínio da informação e suas relações com o projeto de sistemas e programas.

A metodologia de Jackson é especialmente conhecida na Europa e orientada principalmente á aplicações de tempo real.

Segundo Rumbaugh et. al (1994), "o JSD não faz distinção entre a análise e o projeto e em vez disso divide o desenvolvimento de sistemas em duas etapas: especificação e implementação. O JSD determina primeiro *o que fazer* e depois o *como fazer*".

O JSD concentra-se na construção de modelos do domínio de informação do "mundo real". Nas palavras de Jackson (1983): "O desenvolvedor começa criando um modelo da realidade com a qual o sistema está envolvido [...]; é a realidade que fornece seu tema [...]". O autor quis dizer que a realidade é que fornece o tema do sistema.

Segundo Rumbaugh et. al (1994):

> Um modelo JSD começa com uma observação do mundo real. O propósito de um sistema é proporcionar funcionalidade, mas Jackson acha que primeiro se deve considerar como essa funcionalidade se encaixa no mundo real. Um modelo JSD descreve o mundo real em termos de entidades, ações e ordenação de ações. As entidades geralmente aparecem como substantivos nas definições dos requisitos e as ações aparecem como verbos. O desenvolvimento de software JSD consiste em uma seqüência de seis etapas: a etapa de ação da entidade, a de estrutura da entidade, a do modelo inicial, a de função, a de controle do tempo do sistema e a de implementação.

Durante a *etapa de ação da entidade*, o desenvolvedor do software lista as entidades e ações de uma parte do mundo real. O propósito global do sistema orienta a escolha das entidades e das ações. A entrada para a etapa de ação da entidade é a definição dos requisitos (declaração em linguagem natural do problema); enquanto a saída dessa etapa será a lista de entidades e ações, que poderá ser modificada á medida que a análise prosseguir.

Quanto a *etapa de estrutura da entidade* o autor Rumbaugh et. al (1994) afirma que "As ações ocorrem no mundo real, e não são um artefato do sistema. As ações acontecem em um

ponto no tempo, são atômicas e não são passíveis de decomposição. A etapa de estrutura da entidade ordena as ações de cada entidade por tempo".

Segundo Pressman (1995):

> Quando usada no contexto do JSD, a *estrutura* de uma entidade descreve o histórico da entidade ao considerar o impacto das ações ao longo do tempo. Para representar a estrutura da entidade, Jackson introduziu o diagrama de estrutura. O diagrama de estrutura apresenta uma especificação *ordenada quanto ao tempo* das ações executadas em ou por uma entidade. Por essa razão, ele é uma representação mais precisa do mundo real do que uma simples lista de ações e entidades. Um diagrama de estrutura é criado para cada entidade e pode ser acompanhado de um texto narrativo.

A *etapa de modelo inicial* é descrita por Rumbaugh et. al (1994) como segue "A *etapa de modelo inicial* estabelece como o mundo real se interliga com o modelo abstrato".

As etapas de *ação da entidade* e *estrutura da entidade* se preocupam com "uma descrição abstrata do mundo real" (Jackson, 1983). Entidades e ações são selecionadas e relacionadas entre si utilizando o diagrama de estrutura. A etapa de modelo inicial começa a construir uma especificação do sistema como um modelo do mundo real.

Segundo Pressman (1995) "as entidades e ações são representadas como um modelo do processo; as conexões entre o modelo e o mundo real são definidas. A especificação é criada com um *diagrama de especificação do sistema* (System Specification Diagram – SSD)".

As *etapas de função, de controle do tempo* e *de implementação* são descritas por Rumbaugh et. al (1994):

> A *etapa de função* utiliza pseudocódigo para definir as saídas das ações. As funções que correspondem a ações definidas são especificadas. Ao final dessa etapa o analista tem uma especificação completa do sistema solicitado.
> A *etapa do controle do tempo* do sistema avalia qual o retardo que o modelo pode ter em relação ao mundo real. Na maior parte das vezes, o resultado da etapa de controle do tempo é um conjunto de observações informais sobre as restrições de desempenho.
> A *etapa de implementação* focaliza os problemas de escalonamento dos processos e aloca processadores aos processos. O número de processos pode ser diferente do número de processadores. O hardware e o software são especificados como um projeto.
> Após as seis etapas do JSD seguem-se a codificação e o projeto de banco de dados.

Os autores Rumbaugh et. al (1994), considera a abordagem JSD complexa e difícil de compreender integralmente; afirmam que:

> Ela é mais obscura que as abordagens de fluxo de dados e baseada em objetos. Uma razão para a complexidade do JSD é seu forte apoio em pseudocódigo; os modelos gráficos são mais fáceis de compreender. O JSD é complexo também por ter sido projetado especificamente para manipular difíceis problemas de tempo real. Para esses problemas, o JSD pode produzir um projeto superior e valer o esforço. No entanto a complexidade do JSD é desnecessária e um tanto pesada para os problemas mais simples e comuns. O JSD não favorece o amplo entendimento de um problema, tornando a análise de alto nível, ele manipula meticulosamente os detalhes mas não auxilia o desenvolvedor a dominar a essência de um problema. A modelagem JSD é orientada para as ações e distancia-se de entidades e atributos, sendo, desta forma uma técnica pobre para projetos de banco de dados.

2.3.Técnicas Orientadas a Objetos

As Técnicas Orientadas a Objetos começaram a aparecer entre a metade da década de 1970 e o final da década de 1980 quando surgiram as linguagens de programação orientadas a objetos e as pessoas envolvidas com metodologia resolveram experimentar métodos alternativos de análise e projeto.

Sally Shlaer e Steve Mellor escreveram livros sobre análise e projeto orientado a objeto (1988) (1992). Peter Coad e Ed Yourdon (1992) (1993) escreveram livros sobre análise e projeto orientados a objetos. Rumbaugh liderou uma equipe de pesquisadores nos laboratórios da General Eletric, culminando em seu popular livro *Object Modeling Technique* - OMT (1994) que tinha ênfase em análise de sistemas intensivos de dados. Na mesma época Grady Booch (1994) (1995) criou sua técnica voltada para as fases de projeto e construção de sistemas e Ivar Jacobson (1994b) introduziu o conceito de caso de uso através da Object-Oriented Software Engineering (OOSE).

Segundo os autores Rumbaugh, Booch e Jacobson (2000) "a quantidade de técnicas orientados a objetos aumentou de pouco mais de 10 para mais de 50 durante o período de 1989 a 1994". Eles afirmam também que "muitos usuários dessas técnicas tiveram dificuldade para encontrar uma linguagem de modelagem capaz de atender inteiramente às suas necessidades, alimentado, assim, a chamada guerra dos métodos".

Por volta da metade da década de 1990 os três autores Rumbaugh, Booch e Jacobson uniram-se para criar uma linguagem unificada de modelagem, a *Unified Modeling Language*

(UML), proveniente de cada uma de suas técnicas. A unificação das técnicas teve início em outubro de 1994. Em 1996, a UML já era vista pelas organizações como uma ótima estratégia para seus negócios.

Uma força de trabalho foi formada para fazer a padronização da UML na área de metodologias. A versão 1.0 da UML foi oferecida para padronização ao OMG[1] em julho de 1997 e em setembro do mesmo ano foi aceita. Em 14 de novembro de 1997, a UML 1.1 foi adotada como padrão pelo OMG. Neste últimos anos novas revisões foram editadas e apresentadas à comunidade: em julho de 1998 a UML 1.2; no final de 1998 a UML 1.3; no início de 2001 a UML 1.4; na metade de 2001 membros da OMG iniciaram o trabalho de *upgrade* para UML 2.0.

Dentre as Técnicas Orientadas a Objetos que iremos descrever sucintamente nas subseções a seguir, enumeramos:

i. Técnica de Modelagem de Objetos (OMT) de Rumbaugh (1994);
ii. Metodologia de Booch (1994) (1995);
iii. Engenharia de Software Orientada a Objeto (OOSE) de Jacobson (1994b);
iv. Metodologia de Coad e Yourdon (1992) (1993);
v. Unified Modeling Language (UML) de Rumbaugh, Booch e Jacobson (2000).

2.3.1.Técnica de Modelagem de Objetos (OMT)

Conforme Furlan (1998), a Técnica de Modelagem de Objetos (OMT) de Rumbaugh:

> É baseada na modelagem semântica de dados, e tornou-se um enfoque testado e maduro, cobrindo as diversas fases do desenvolvimento orientado a objetos. A notação empregada por Rumbaugh é parecida com a dos métodos estruturados e utiliza a notação de modelo de objeto que suporta conceitos de modelagem de dados (exemplo: atributos e relacionamentos), objetos (composição/ agregação) e herança. Também é empregada para categorizar classes e instâncias. O ponto forte do método Rumbaugh é a notação utilizada e o enfoque relativamente conservador no uso da teoria de objetos. Por outro lado, um problema apresentado é a falta de notação específica para representar a passagem de mensagem de um objeto a outro.

[1] A OMG é uma organização internacional, que promove a teoria e prática das técnicas orientadas a objeto em desenvolvimento de sistemas. A patente da organização inclui o estabelecimento de normas industriais e especificações de gerenciamento de objetos para prover um padrão comum para o desenvolvimento de aplicações.

A OMT é uma metodologia que combina três visões da modelagem de sistemas:

i. *o modelo de objetos* que descreve os objetos do sistema e seus relacionamentos;

ii. *o modelo dinâmico* que descreve as interações entre os objetos do sistema;

iii. *o modelo funcional* que descreve as transformações de dados do sistema.

Cada modelo é aplicável durante todas as etapas do desenvolvimento sendo-lhes adicionados detalhes de implementação à proporção que o desenvolvimento progride. A descrição completa de um sistema exige todos os três modelos.

O modelo de objetos mostra a estrutura estática do mundo real. São identificadas as classes de objetos, as associações entre objetos, os atributos e as ligações. As informações contidas no modelo de objetos são suplementadas por descrições textuais breves, incluindo o propósito e o escopo de cada entidade.

O modelo dinâmico mostra o comportamento do sistema, principalmente a seqüência de interações. São preparados os cenários de sessões típicas e excepcionais. São identificados os eventos externos entre o sistema e o mundo exterior. É elaborado um diagrama de estados para cada objeto ativo. São comparados os eventos entre diagramas de estado.

O modelo funcional mostra a derivação funcional de valores, sem preocupações sobre quando eles são calculados. Primeiro são identificados os valores de entrada e saída do sistema como parâmetros de eventos externos. Em seguida, são construídos diagramas de fluxo de dados para mostrar o processamento de cada valor de saída a partir de outros valores e, em última análise, dos valores de entrada. Por fim, são especificadas as restrições e os critérios de otimização.

Rumbaugh et. al (1994) descreve:

> Na OMT primeiro prepara-se um modelo que sumarize os aspectos essenciais do domínio da aplicação, sem preocupações com uma eventual implementação. Esse modelo contém objetos

encontrados no domínio da aplicação, incluindo uma descrição das propriedades dos objetos e de seu comportamento. Em seguida, são tomadas decisões acerca do projeto e acrescentam-se detalhes ao modelo para se descrever e otimizar a implementação. Os objetos do domínio da aplicação compõem a estrutura do modelo projetado, mas são implementados em termos de objetos do domínio do computador. No final, o modelo projetado é implementado em uma linguagem de programação, em um banco de dados ou em hardware.

A Metodologia OMT compõem-se das fases:

i. análise;

ii. projeto do sistema;

iii. projeto dos objetos;

iv. implementação.

A *análise* preocupa-se em definir e permitir o entendimento do problema e do domínio da aplicação para que se possa construir um projeto correto. Essa fase incorpora as características essenciais do problema sem introduzir aspectos da implementação para que não sejam restringidas prematuramente as decisões de projeto. A entrada inicial da fase de análise é um enunciado que descreve o problema a ser solucionado e oferece uma visão geral conceitual do sistema proposto. Outras entradas para a análise são entrevistas com o usuário e conhecimento do ambiente do mundo real. A saída da etapa de análise é um modelo formal que incorpora os três aspectos essenciais do sistema: os objetos e o relacionamento entre eles; o fluxo dinâmico de controle; e a transformação funcional dos dados sujeita a restrições. O resultado da análise substitui o enunciado original do problema e serve de base para o projeto. Durante a análise, o enfoque é sobre *o que* precisa ser feito, independente de como deve ser feito.

Na fase de *projeto do sistema*, é decidido *como* o problema será resolvido, primeiro em alto nível e depois em níveis cada vez mais detalhados. Nesta fase a arquitetura geral do sistema é definida. Utilizando-se o modelo de objetos como guia, o sistema é organizado em subsistemas. A concorrência é organizada agrupando-se os objetos em tarefas concorrentes. São tomadas decisões gerais sobre comunicações entre os processos, armazenamento de

dados e implementação do modelo dinâmico. As prioridades são estabelecidas para serem feitos ajustes no projeto.

Na fase de *projeto dos objetos*, são elaborados, refinados e otimizados os modelos de análise para a produção de um projeto prático. Durante esta fase ocorre um deslocamento na ênfase dos conceitos de aplicações em direção aos conceitos computacionais. Primeiro são escolhidos os algoritmos básicos para a implementação das principais funções do sistema. Com base nesses algoritmos, a estrutura do modelo de objetos é otimizada com vistas a uma implementação eficiente. O projeto também deve levar em conta a concorrência e o fluxo dinâmico de controle como determinado durante o projeto do sistema. Por fim, os subsistemas são empacotados em módulos.

A fase de *implementação* é uma extensão das fases de projeto do sistema e projeto de objetos. Ela é direta, quase mecânica, porque todas as decisões difíceis já foram tomadas. A implementação deve ser uma simples tradução das decisões das fases de projeto nas particularidades de uma linguagem de programação específica. É claro que é preciso tomar decisões durante a implementação, mas as decisões tomadas nesta fase afetam somente uma pequena parte do código do programa. Contudo, o código do programa é a corporificação definitiva da solução do problema e, portanto, a maneira como ele é escrito tem grande importância para que se possa obter facilidade de manutenção e ampliação.

2.3.2.Metodologia de Booch

Grady Booch (1994) propôs um método que consiste no emprego de técnicas de projeto orientado a objeto, apesar de ter sido estendido para contemplar também a análise orientada a objeto.

Segundo Furlan (1998) “Booch descreve um objeto como sendo um modelo do mundo real que consiste de dados e habilidades para o tratamento desses dados. Uma vez tendo sido

localizados e definidos os objetos, esses passam a servir de base para os módulos do sistema ou são considerados módulos relacionados".

Furlan (1998) afirma que:

> O projeto orientado a objeto, na metodologia de Booch, é organizado via abstrações algorítmicas de forma parecida à programação orientada a objeto. Cada módulo representa um objeto ou uma classe de objetos com reutilização e encapsulamento ao longo dos processos. Essa estreita relação existente entre projeto e programação orientados a objeto permite aos projetistas tomar decisões de negócio antes da criação do código, contribuindo para a melhoria do sistema gerado.

Segundo Booch (1994) sua metodologia usa os seguintes tipos de diagramas para descrever as decisões de análise e projeto, que devem ser utilizados na criação de um sistema orientado a objetos:

i. Diagrama de classes;
ii. Diagrama de objetos;
iii. Diagramas de estado;
iv. Diagrama de módulos;
v. Diagrama de processos;
vi. Diagrama de interação.

O *Diagrama de classes* consiste em um conjunto de classes e relacionamentos entre elas. Pode conter classes, classes parametrizadas, utilidades e metaclasses. Os tipos de relações são associações, herança, instância, entre outras.

O *Diagrama de objetos* mostra os objetos e seus relacionamentos. Enquanto que os relacionamentos entre as classes são estáticos, os relacionamentos entre os objetos são dinâmicos.

O *Diagrama de estado* mostra como as instâncias de classes se movem de um estado para o outro, sob a influência dos eventos, e quais ações resultam essas mudanças de estados.

O *Diagrama de módulos* mostra o alocamento físico das classes aos módulos.

O *Diagrama de processo* mostra o alocamento dos processos aos processadores na vista de um sistema.

O *Diagrama de interação* mostra o mesmo que diagrama de objeto, mas de outra maneira. É recomendado usar diagramas de interação para trabalhar em um ambiente em tempo real.

A visão geral dos passos do processo da metodologia de Booch é:

i. identificação de classes e objetos;
ii. identificação da semântica das classes e objetos;
iii. identificação dos relacionamentos entre classes e objetos;
iv. implementação das classes e objetos.

O primeiro passo consiste na identificação dos objetos e das classes que formam parte do sistema. Em seguida, constrói-se a interface das classes ("identificação da semântica"), seguindo-se a descoberta dos relacionamentos existentes entre elas. A descoberta dos relacionamentos pode provocar a criação de novas interfaces, de forma que esses dois passos podem ser executados internamente até que seja obtido um estado satisfatório. Neste ponto, o estágio de implementação decide sobre os aspectos de representação interna das classes (atributos e comportamento). Isso pode resultar em termos que aplicar todo o processo novamente ao comportamento de uma única classe.

Na etapa de *Análise* se define o que os usuários do sistema desejam. É uma etapa de alto nível que identifica as principais funções do sistema, o que será modelado do domínio do problema e a documentação dos processos principais e as políticas que o sistema deve suportar. Não são definidos passos formais já que esses dependem de quão novo é o projeto, da disponibilidade de especialistas, usuários e de documentos adicionais.

Ainda na etapa de Análise são vistas as principais entradas e saídas do sistema, referências a políticas, procedimentos dos sistemas existentes, etc. Existe um conjunto de

mecanismos chaves que o sistema deve prover são o estado da entrada; o estado da saída; e os estados esperados. Na etapa de Análise o processo é definido de forma concisa, precisa, e a parte do modelo do domínio do sistema é orientada a objeto.

Na etapa de *Projeto*, é determinada a implementação efetiva do processo e a eficiência da realização das funções através da informação da análise do problema.

2.3.3.Engenharia de Software Orientada a Objeto (OOSE)

Jacobson (1994b) é o criador do método Object Oriented Software Engineering (OOSE). Sua abordagem considera que a propriedade mais essencial de um sistema é estabelecer a estrutura (arquitetura) durante o ciclo de vida. Ele afirma, que a melhor forma de assegurar isso é modelar as funcionalidades do sistema na base da organização e seus usuários.

O OOSE pertence a um número crescente de métodos que consideram modelar o comportamento, ou uma descrição de alto nível das funcionalidades do sistema do ponto de vista de seus usuários, como o centro do desenvolvimento do sistema.

No OOSE a descrição de alto nível do comportamento é baseada nos *atores* e *casos de uso*. Esses conceitos ajudam na definição de quais saídas existem no sistema – os atores – e o que deve ser realizado pelo sistema – os casos de uso.

Desenvolvendo um sistema dessa forma, a arquitetura do sistema inteiro será controlada pelo que os usuários desejam fazer com o sistema. O uso da metodologia orientada a objeto para modelar sistemas assegura também que todos os modelos serão traçados através dos requisitos para a codificação e manutenção. Em combinação com a abordagem do caso de uso, isso permitirá a modificação do sistema e a inserção dos novos requisitos que surgirem.

Segundo Jacobson (1994b):

> O modelo com *atores* mostra a perspectiva dos usuários – tudo o que é externo que se comunica com o sistema, incluindo outros sistemas. Atores representam um determinado papel, ou descrição

> de classe do comportamento, desde que exista uma pessoa usando realmente o sistema. Eles são a ferramenta principal para encontrar *casos de uso*, e juntos *atores* e *casos de uso* definem completamente a funcionalidade do sistema. Cada *caso de uso* é um curso completo dos eventos no sistema na perspectiva dos usuários. *Casos de uso* é a linha central de funcionamento do OOSE.

OOSE vê o desenvolvimento do sistema através da construção de modelos. Isto inclui três processos principais de desenvolvimento: análise, construção e testes. Cada um desses processos produz um ou mais modelos associados. O processo de *análise* produz o modelo de requisitos e o modelo de análise. O processo de *construção* produz o projeto e o modelo de implementação. Enquanto o processo de *testes* produz o modelo de testes.

O processo OOSE especifica uma série de modelos como produtos no processo de construção do sistema, constituindo uma fase dentro do processo iterativo total. São eles:

i. modelo de requisitos;

ii. modelo de análise;

iii. modelo de projeto;

iv. modelo de implementação;

v. modelo de teste.

O *modelo de requisitos* é o ponto de delimitação do sistema e define quais funcionalidades o sistema deverá oferecer em termos de atores e casos de uso. Isto deverá servir como um elo de comunicação entre os desenvolvedores e o ordenador do sistema.

Quando o modelo de requisitos inicial é estabelecido e aprovado pelos usuários ou ordenadores do sistema, o desenvolvimento do sistema real inicia pelo desenvolvimento do *modelo de análise*. Assim, as estruturas modeladas do sistema são independentes do ambiente real de implementação. Isso deriva do modelo de requisitos e forma a base da arquitetura do sistema.

O *modelo de projeto* é um refinamento e formalização do modelo de análise. Isto é, o primeiro modelo que leva em conta o real ambiente de implementação do sistema. Neste

modelo, a interface dos objetos e a semântica de suas operações são explicitamente definidas. Outro ponto importante do ambiente do sistema é assegurado pelo DBMS, características da linguagem de programação e distribuição.

O *modelo de implementação* consiste do código fonte composto ou escrito para o sistema. Ele implementa cada objeto específico especificado nos modelos acima.

O *modelo de teste* suporta a verificação do sistema. Ele envolve principalmente a documentação das especificações de teste e dos resultados dos testes. Os teste iniciam nos níveis mais baixos (testando unidade) e prossegue nos *casos de usos* e finalmente no sistema inteiro (testes de integração).

O desenvolvimento desses modelos é iterativo, e os modelos serão submetidos a muitas mudanças até se tornarem estáveis. Geralmente, o trabalho de modelagens sucessivas não é iniciado até que os resultados de modelagens de fases prévias estejam estabilizados. Entretanto, o esboço dos modelos nas fases subseqüentes, a revisão e o refinamento dos modelos das fases prévias, algumas vezes, tornar-se-á necessário para o trabalho da fase atual. Cada uma dessas fases de modelagem tem um método associado a ela, o qual compreende certos passos a serem tomados na construção e refinamento do modelo. O processo dentro de cada fase não é estritamente linear, e o método para a fase de modelagem é tipicamente aplicado de forma iterativa.

Segundo Furlan (1998):

> O que diferencia o método de Jacobson de outros métodos orientados a objetos é o seu foco em casos de uso e a categorização de pessoas e equipamentos dependendo de seu papel no sistema global. A análise no método de Jacobson é baseada em modelos de requerimentos e análise que consistem de um conjunto de casos de uso, de um modelo de domínio de problema e de uma descrição da interface do sistema. Seguindo a criação do modelo de análise, são gerados diagramas baseados no modelo que também descrevem a comunicação entre blocos. O modelo de blocos é então implementado utilizando-se linguagens apropriadas e ferramentas orientadas a objeto. Um dos pontos fracos do enfoque original da OOSE de Jacobson é a notação simplista usada para objetos de domínio (objetos simplesmente não representam como círculos). Por outro lado, o método Objectory tem sido adaptado para a engenharia de negócio, onde as idéias são usadas para modelar e melhorar processos.

2.3.4.Metodologia Coad/Yourdon

Peter Coad e Ed Yourdon (1992) (1993), com seu enfoque simples, porém eficaz, dividiram a análise orientada a objeto como sendo classes e objetos. Os objetos são abstrações de um domínio de problema que reflete a habilidade de um sistema em reter dados, interagir com dados e encapsular valores de atributos e serviços. Classe é uma coleção de um ou mais objetos com atributos e serviços, abrangendo o entendimento de como um novo objeto é criado em uma classe.

Segundo Coad (1992) “a OOA (Análise Orientada a Objeto) identifica e define Classes e Objetos que refletem diretamente o domínio do problema e as responsabilidades do sistema dentro dele”.

Segundo Coad (1993) “o OOD (Projeto Orientado a Objeto) identifica e define Classes e Objetos adicionais, refletindo uma implementação dos requisitos”.

A OOA e o OOD consistem de duas atividades distintas – que são aplicadas em seqüência ou alguma espécie de entrelaçamento.

O modelo do OOD, exatamente como o modelo da OOA, consiste em cinco camadas: Assunto, Classe e Objeto, Estrutura, Atributo e Serviço. Estas cinco camadas são muito semelhantes a transparências que, quando colocadas umas sobre as outras apresentam gradualmente mais e mais detalhes. As camadas são cortes horizontais do modelo geral.

Numa abordagem global, as cinco camadas correspondem às cinco atividades principais introduzidas na OOA, são elas:

i. determinar classes e objetos;
ii. identificar estruturas;
iii. identificar assuntos;
iv. definir atributos;
v. definir serviços.

Para OOA, os termos “objeto” e “classe” são definidos de forma a exprimir o domínio do problema e as responsabilidades do sistema. Na identificação das Classes e Objetos a intenção é traçar um paralelo entre a representação técnica de um sistema e a visão conceitual do mundo real.

Para Coad (1992), “Estrutura é uma representação de um domínio de problema que está diretamente relacionado às competências do sistema”. O termo “Estrutura” é usado como um termo global, podendo ser usado para Estrutura de Generalização-Especialização (Gen-Espec) e Estrutura Todo-Parte. A Estrutura Gen-Espec pode ser usada para a “distinção entre Classes”. Nas Estruturas Gen-Espec pode ser aplicada a herança, com uma representação explícita de Atributos e Serviços mais gerais seguidos por especializações pertinentes. A Estrutura Todo-Parte é um dos três métodos básicos de organização naturais dos seres humanos.

Um Assunto é um mecanismo para orientar um leitor (analista, especialista de domínio de problemas, gerente, cliente) em um modelo amplo e complexo. Os Assuntos também são úteis para a organização de pacotes de trabalho em projetos extensos, conforme as investigações iniciais de OOA. Os Assuntos têm um objetivo específico – proporcionar uma visão geral de um modelo extenso de OOA. A base principal para a identificação de Assuntos é a complexidade do domínio do problema, como identificada pelas Estruturas Gen-Espec e Estruturas Todo-Parte. Desta forma, os Assuntos são *partes* usadas para comunicar o *todo* de um domínio.

Coad (1992) afirma que “um Atributo é um dado (informação de estado) para o qual cada Objeto em uma Classe tem seu próprio valor”. Os Atributos adicionam detalhes às abstrações “Classe e Objeto” e “Estrutura”. Os Atributos descrevem valores (estado) mantidos em um Objeto, que devem ser manipulados exclusivamente pelos Serviços deste Objeto.

Conforme Coad (1992) “um Serviço é um comportamento específico que um Objeto deve exibir”. Definir Serviços é definir a comunicação necessária entre Objetos. Tais comandos e solicitações fazem parte da natureza da interação humana com um sistema. E o mesmo modelo de interação é usado entre as partes do modelo OOA.

Segundo Coad (1993) “o modelo do OOD consiste em quatro componentes: Componente Domínio do Problema; Componente Interação Humana; Componente Gerenciamento de Tarefas; Componente Gerenciamento de Dados”. Os componentes são cortes verticais do modelo geral.

Numa abordagem global, os quatro componentes correspondem às quatro atividades principais no OOD, a saber:

i. projetar o componente domínio do problema;

ii. projetar o componente interação humana;

iii. projetar o componente gerenciamento de tarefas;

iv. projetar o componente gerenciamento de dados.

No projeto, os resultados da análise se encaixam exatamente no *componente domínio do problema*. Os resultados da análise são uma parte essencial do modelo multicomponentes do projeto. Esses resultados da análise são aperfeiçoados e acrescidos durante o projeto.

A *interação humana* precisa de exame detalhado, tanto na análise como no projeto. Na análise, esse exame detalhado é feito de modo que os *atributos* e *serviços* requeridos sejam especificados. A prototipação é usada para auxiliar durante a extração e especificação dos requisitos. No projeto, o componente interação humana acrescenta a esses resultados o projeto da interação humana e os detalhes dessa interação. Isso inclui o formato desenhado de janelas e relatórios. A prototipação é utilizada para auxiliar no desenvolvimento e seleção dos mecanismos reais de interação.

Para certas aplicações, as tarefas simplificam o projeto e o código globais. Diferentes *tarefas* separam comportamentos que devem ocorrer concorrentemente. O ponto principal desta estratégia é identificar e projetar as *tarefas* e os *serviços* incluídos em cada tarefa.

O *componente* g*erenciamento de dados* fornece a infra-estrutura para o armazenamento e a recuperação de objetos de um sistema de gerenciamento de dados.

2.3.5.Unified Modeling Language (UML)

A UML não é um método, é uma linguagem para modelagem de sistemas orientados a objetos. Os métodos, na sua maioria, são compostos de uma linguagem de modelagem (notação gráfica) e de um processo (passos para a elaboração de um projeto).

A UML é uma linguagem para especificação, visualização, construção e documentação de artefatos de sistemas de software. Exemplos de artefatos: requisitos, arquitetura, projeto, código-fonte, planos do projeto, testes, protótipos, versões, etc.

Rumbaugh, Booch e Jacobson (2000) acrescentam:

> A UML proporciona uma forma padrão para a preparação de planos de arquitetura de projetos de sistemas, incluindo aspectos conceituais tais como processos de negócios e funções do sistema, além de itens concretos como as classes escritas em determinada linguagem de programação, esquemas de bancos de dados e componentes de software reutilizáveis.

Quando falamos em *especificação*, dizemos que a UML através de modelos precisos, completos e sem ambigüidades, atende às decisões que necessitam ser tomadas para o desenvolvimento e implantação de sistemas.

No desenvolvimento da maioria dos sistemas precisamos de um padrão para modelagem gráfica, a fim de que um desenvolvedor possa escrever seu modelo e qualquer outro possa interpretá-lo sem ambigüidades. A UML através de sua semântica bem definida, atende plenamente a essa necessidade. Essa *visualização* precisa dá aos desenvolvedores e às ferramentas condições de interpretar os modelos sem ambigüidades.

A UML auxilia na *construção* de software pois permite um mapeamento de seus modelos para as linguagens de programação, ou vice-versa. Assim, é possível gerar código a partir de modelos da UML, ou com suporte adequado realizar a engenharia reversa, reconstruindo um modelo a partir de sua implementação.

A UML facilita a *documentação* pois possui suporte para criação e documentação de vários artefatos que são gerados durante o desenvolvimento de um sistema.

O projeto da UML procurou desenvolver uma linguagem de modelagem que atingisse as seguintes metas:

i. prover à comunidade uma linguagem de modelagem visual pronta para o uso e expressiva, possibilitando desenvolver e intercambiar modelos significativos;
ii. fornecer extensibilidade e mecanismos de especialização para estender os conceitos centrais;
iii. suportar especificações que são independentes de processos de desenvolvimento e linguagens de programação particulares;
iv. prover uma base formal para entendimento da linguagem de modelagem;
v. encorajar o crescimento do mercado de ferramentas de objetos;
vi. suportar alto nível de conceitos de desenvolvimento como componentes, colaborações, estruturas e padrões;
vii. integrar melhores práticas.

A UML alcançou dois aspectos muito importantes. Primeiro, terminou com as diversas diferenças existentes entre os métodos de modelagem anteriores. Segundo, unificou as perspectivas entre muitos sistemas de tipos diferentes (negócio x software), fases de desenvolvimento (análise de requisitos, projeto e implementação) e conceitos internos. Por esses aspectos, é essa a linguagem de modelagem escolhida para desenvolvemos o estudo de caso deste trabalho.

CONSIDERAÇÕES

As técnicas ou métodos de modelagem de sistemas de informação oferecem uma abordagem sistemática para análise de problemas. Embora cada técnica tenha um conjunto único de procedimentos e simbologias, todas oferecem mecanismos para avaliação e representação do domínio da informação.

As Técnicas Estruturadas salientam a decomposição funcional, nelas as funções são mais importantes que os dados. A notação original da Técnica estruturada foi desenvolvida para aplicações de processamento de dados convencional, mas extensões foram criadas para que a técnica fosse aplicada a sistemas de tempo real.

As Técnicas Orientadas à Estrutura de Dados não foram tão amplamente usadas como as Técnicas Estruturadas, mas têm muitos aspectos em comum com elas. É importante dizer que cada uma introduz sua própria visão em particular da modelagem de sistemas. As Técnicas Orientadas à Estrutura de Dados identificam os itens de informação e as ações (processos) e modelam-nos de acordo com a hierarquia (estrutura) de informações do problema.

As Técnicas Orientadas a Objetos por sua vez vêem o mundo como uma coletânea de objetos que interagem entre si, apresentam características próprias que são representadas pelos seus atributos (dados) e operações (processos). As operações de processamento fazem parte do objeto e são iniciadas ao passar uma mensagem ao objeto.

Um aspecto importante das Técnicas Orientadas a Objetos é a sua característica intrínseca de analisar o mundo como ele é, permitindo organizar resultados de maneira mais fácil e natural.

Técnicas estruturadas	**Vantagens** • conseguiram melhorar o processo de desenvolvimento, na época que foram criadas, fazendo com que o desenvolvimento fosse enxergado com mais esmero, respeitando sua complexidade;

<table>
<tr><td></td><td>Desvantagens
há dificuldade em garantir compatibilidade entre as fases de análise e projeto e posteriormente do projeto para implementação;
quando o limite do sistema que utiliza esta técnica é estabelecido, a ampliação do projeto torna-se difícil para um novo limite, devido a estrutura do projeto ser derivada do limite inicial estabelecido;
há dificuldade em reutilizar componentes de um projeto, no projeto seguinte, devido a decomposição funcional ser baseada no processo, pessoas diferentes tendem a produzir decomposições diferentes com o uso desta técnica;
na maioria das vezes, grandes alterações ou extensões dos modelos criados geram um grande esforço;
a comunicação entre desenvolvedores e usuários é difícil, em virtude dos diagramas não serem muito expressivos;
validações dos usuários dificultadas pelas ferramentas usadas nessa técnica.</td></tr>
<tr><td rowspan="2">Técnicas alternativa de análise orientada a estrutura de dados</td><td>Vantagens
para alguns tipos de aplicações (softwares concorrentes, software de tempo real, microcódigo, etc.) são metodologias úteis.</td></tr>
<tr><td>Desvantagens
a maioria delas é de difícil compreensão integralmente pois foram projetadas para problemas específicos;
inadequadas para algumas aplicações (análise de alto risco, projetos de bancos de dados, softwares convencionais);
não foram tão amplamente usadas;</td></tr>
<tr><td rowspan="2">Técnicas orientadas a objeto</td><td>Vantagens
projetos baseados em objetos são mais elásticos quanto a modificações e mais extensíveis, bastando apenas acrescentar objetos e relacionamentos;
a analogia direta entre objetos de um projeto baseado em objetos do domínio do problema resulta em sistemas mais fáceis de compreender. Isso torna o projeto mais intuitivo e simplifica o rastreamento entre requisitos e o código do software;
há facilidade em reutilizar componentes de um projeto no projeto seguinte, devido a decomposição funcional ser baseada em objetos, pessoas diferentes tendem a identificar objetos semelhantes com o uso desta técnica;
uniformizaram os modelos usados para análise, projeto e implementação. Os principais diagramas são utilizados em todas as fases mudando-se apenas a sua visão;
a comunicação entre usuários e desenvolvedor tornou-se mais fácil, pela análise e validação dos diagramas apresentados;
alguns autores afirmam que a orientação a objetos veio propiciar aumento de produtividade, diminuição do custo de desenvolvimento e manutenção, maior portabilidade e reutilização de código.</td></tr>
<tr><td>Desvantagens
o paradigma precisa evoluir mais, e ser mais utilizado, ainda é considerado relativamente novo.</td></tr>
</table>

Tabela 1 – Quadro comparativo das técnicas estudadas.

O quadro comparativo, tabela 1, mostra vantagens e desvantagens das técnicas estudadas neste capítulo.

Na subseção sobre a UML procuramos dar apenas uma visão global do que vem a ser a Linguagem de Modelagem Unificada. No próximo capítulo vamos detalhar como modelamos sistemas com a UML. Veremos os seus elementos fundamentais como: blocos de construção; regras de formatação; mecanismos básicos e, para finalizar, como a arquitetura de um sistema é expressa por ela.

CAPÍTULO 3 – UNIFIED MODELING LANGUAGE (UML)

A UML é adequada para a modelagem de sistemas, cuja abrangência poderá incluir sistemas de informação corporativos a serem distribuídos a aplicações baseadas em Web (World Wide Web) e até sistemas complexos embutidos de tempo real. É uma linguagem muito expressiva, abrangendo todas as visões necessárias ao desenvolvimento e implantação desses sistemas. Sua aplicação tem início com a formação de um modelo conceitual da linguagem, o que pressupõe o entendimento de três principais elementos: os blocos básicos de construção da UML, as regras que determinam como esses blocos de construção deverão ser combinados e alguns mecanismos básicos que se aplicam a toda a linguagem (RUMBAUGH et al, 2000).

3.1. Blocos de construção da UML

O vocabulário da UML abrange três tipos de blocos de construção: itens, relacionamentos e diagramas.

Os itens são as abstrações identificadas como cidadãos de primeira classe em um modelo; os relacionamentos reúnem esses itens; os diagramas agrupam coleções interessantes de itens.

3.1.1. Itens da UML

Existem quatro tipos de itens na UML: itens estruturais, itens comportamentais, itens de agrupamentos e itens anotacionais. Estes itens constituem os blocos de construção básicos orientados a objetos da UML e são utilizados para escrever os modelos bem-formados.

3.1.1.1.Itens Estruturais

Os itens estruturais são os substantivos utilizados em modelos da UML. São as partes mais estáticas do modelo, representando elementos conceituais ou físicos. Ao todo existem sete tipos de itens estruturais.

Primeiro, *as classes* são descrições como conjuntos de objetos que compartilham os mesmos atributos, operações, relacionamentos e semântica. As classes implementam uma ou mais interfaces. Graficamente, as classes são representadas por retângulos, geralmente incluindo o seu nome, atributos e operações, conforme mostra a Figura 1.

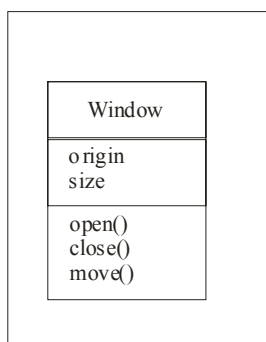



Figura 1 – Classes

Segundo, uma *interface* é uma coleção de operações que especificam serviços de uma classe ou componente. Portanto, uma interface descreve o comportamento externamente visível desse elemento. Uma interface poderá representar todo o comportamento de uma classe ou componente, como também apenas parte desse comportamento. A interface define um conjunto de especificações de operações (suas assinaturas), mas nunca um conjunto de implementações de operações. A interface é representada graficamente como um círculo e o respectivo nome. Uma interface raramente aparece sozinha. Em vez disso, costuma aparecer anexada à classe ou ao componente que realiza a interface, conforme mostra a Figura 2.

Figura 2 – Interface

Terceiro, *as colaborações* definem interações e são sociedades de papéis e outros elementos que funcionam em conjunto para proporcionar um comportamento cooperativo superior à soma de todos os elementos. Uma colaboração também é a especificação do modo como um elemento, tal como um classificador (incluindo uma classe, interface, componente, nó ou caso de uso) ou uma operação, é realizada por um conjunto de classificadores e associações desempenhando papéis específicos, utilizados de uma maneira específica. Graficamente, as colaborações são representadas como elipses com linhas tracejadas, geralmente incluindo somente seu nome, conforme mostra a Figura 3.

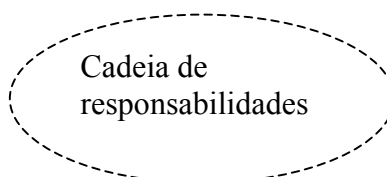



Figura 3 - Colaborações

Quarto, um *caso de uso* é a descrição de um conjunto de seqüência de ações realizadas pelo sistema que proporciona resultados observáveis de valor para um determinado ator. Um caso de uso é utilizado para estruturar o comportamento de itens de um modelo. Um caso de uso é realizado por uma colaboração. Graficamente, um caso de uso é representado por uma elipse com linhas contínuas, geralmente incluindo somente seu nome, conforme mostra a Figura 4.

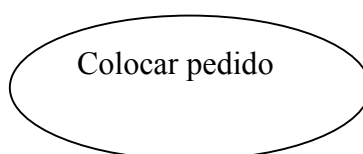



Figura 4 - Caso de Uso

Quinto, *as classes ativas* são classes cujos objetos têm um ou mais processos ou threads[2] e, portanto, podem iniciar a atividade de controle. Uma classe ativa é semelhante a uma classe, exceto pelo fato de que seus objetos representam elementos cujo comportamento é concorrente com o de outros elementos. Graficamente, as classes ativas são representadas da mesma maneira como as classes, mas com linhas mais grossas, incluindo seus nomes, atributos e operações, conforme mostra a Figura 5.

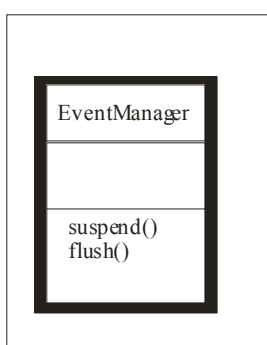



Figura 5 - Classes Ativas

Os dois elementos restantes (componentes e nós) representam itens físicos, enquanto os cinco itens anteriores representam itens conceituais ou lógicos.

Sexto, os *componentes* são partes físicas e substituíveis de um sistema, que proporcionam a realização de um conjunto de interfaces. Em um sistema encontram-se diferentes tipos de componentes próprios da implantação, como os componentes COM+ ou Java Beans, assim como componentes que são artefatos do processo de desenvolvimento, como os arquivos de código-fonte. Tipicamente os componentes representam o pacote físico de elementos lógicos diferentes, como classes, interfaces e colaborações. Graficamente, os componentes são representados como retângulos com abas, incluindo somente seus nomes, conforme mostra a Figura 6.

---

[2] thread um fluxo leve de controle que pode ser executado concorrentemente com outros threads no mesmo processo.

Sétimo, um *nó* é um elemento físico existente em tempo de execução que representa um recurso computacional, geralmente com pelo menos alguma memória e, freqüentemente, capacidade de processamento. Um conjunto de componentes poderá estar contido em um nó e também poderá migrar de um nó para outro. Graficamente, um nó é representado como um cubo, normalmente incluindo somente seu nome, conforme mostra a Figura 7.

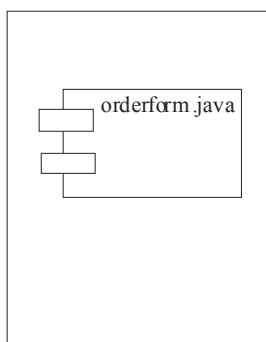



Figura 6 - Componentes

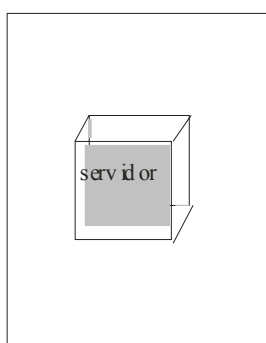



Figura 7 - Nó

Esses sete elementos (classes, interfaces, colaborações, casos de uso, classes ativas, componentes e nós) são os itens estruturais básicos que poderão estar incluídos em um modelo da UML. Também existem variações desses sete elementos, como atores, sinais e utilitários (tipos de classes), processos e threads (tipos de classes ativas), e aplicações, documentos, arquivos, bibliotecas, páginas e tabelas (tipos de componentes).

3.1.1.2.Itens Comportamentais

Os itens comportamentais são as partes dinâmicas dos modelos UML. São os verbos de um modelo, representando comportamentos no tempo e no espaço. Ao todo, existem dois tipos principais de itens comportamentais: interação e máquina de estado.

Primeiro, uma *interação* é um comportamento que abrangem um conjunto de mensagens trocadas entre um conjunto de objetos em determinado contexto para a realização de propósitos específicos. O comportamento de uma sociedade de objetos ou de uma operação individual poderá ser especificado por meio de uma interação. As interações envolvem outros elementos, inclusive mensagens, sequencias de ações (os comportamentos chamados pelas mensagens) e ligações (as conexões entre os objetos). Graficamente, uma mensagem é representada como linha cheia com seta, quase sempre incluindo o nome de suas operações, conforme mostra a Figura 8.

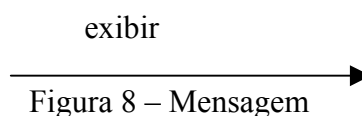
Figura 8 – Mensagem

Segundo, uma *máquina de estado* é um comportamento que especifica as sequencias de estados pelas quais objetos ou interações passam durante sua existência em resposta a eventos, bem como suas respostas a esses eventos. O comportamento de uma classe individual ou de uma colaboração de classes pode ser especificado por meio de uma máquina de estados. Uma máquina de estado abrange outros elementos, incluindo estados, transições (o fluxo de um estado a outro), eventos (itens que disparam uma transição) e atividades (as respostas às transições). Graficamente, o estado é representado como retângulo com ângulos arredondados, geralmente incluindo seu nome e respectivos subestados, se houver, conforme mostra a Figura 9.

Figura 9 – Estado

Esses dois elementos (interações e máquinas de estados) são os itens comportamentais básicos que podem estar incluídos em um modelo da UML. Semanticamente, esses elementos costumam estar conectados a vários elementos estruturais, classes principais, colaborações e objetos.

3.1.1.3.Itens de Agrupamentos

Os itens de agrupamentos são as partes organizacionais dos modelos de UML. São os blocos em que os modelos podem ser decompostos. Ao todo, existe apenas um tipo principal de itens de agrupamento, chamado *pacotes*.

Um *pacote* é um mecanismo de propósito geral para a organização de elementos em grupos. Os itens estruturais, os itens comportamentais e até outros itens de grupos podem ser colocados em pacotes. Ao contrário dos componentes (que existem em tempo de execução), um pacote é puramente conceitual (o que significa que existe apenas em tempo de desenvolvimento). Graficamente, um pacote é representado como diretórios com guias, geralmente incluindo somente seus nomes e, às vezes, seu conteúdo, conforme mostra a Figura 10.

Os pacotes são itens de agrupamento básico, com os quais se pode organizar modelos de UML. Também existem variações, como frameworks[3], modelos e subsistemas (tipos de pacotes).

---

[3] framework - um padrão de arquitetura que fornece um template extensível para aplicações em um domínio. Template - um elemento parametrizado.

Figura 10 – Pacote

3.1.1.4.Itens Anotacionais

Os itens anotacionais são as partes explicativas de um modelo UML. São comentários, incluídos para descrever, esclarecer e fazer alguma observação sobre qualquer elemento do modelo. Existe um único tipo de item anotacional, chamado nota. Uma *nota* é apenas um símbolo para representar restrições e comentários anexados a um elemento ou a uma coleção de elementos. Graficamente, uma nota é representada por um retângulo com um dos cantos como uma dobra de página, acompanhado por texto ou comentário gráfico, conforme mostra a Figura 11.

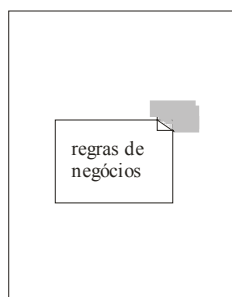



Figura 11 - Nota

Esse elemento é o único item anotacional básico que poderá ser incluído em um modelo de UML. Geralmente as notas são usadas para aprimorar os diagramas e restrições ou comentários que possam ser mais bem expressos por um texto formal ou informal. Também existem variações desse elemento, como os requisitos (que especificam determinado comportamento desejado sob uma perspectiva externa ao sistema).

3.1.2. Relacionamentos na UML

Existem quatro tipos de relacionamentos na UML: dependência, associação, generalização e realização. Esses relacionamentos são os blocos relacionais básicos de construção da UML. São utilizados para escrever modelos bem-formados.

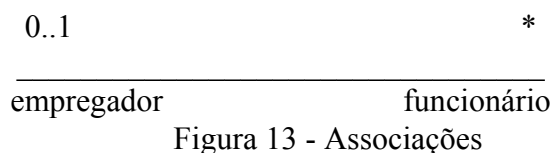
Primeiro, uma *dependência* é um relacionamento de utilização, determinando as modificações na especificação de um item (por exemplo, a classe *Evento)*, pode afetar outro item que a utilize (por exemplo, a classe *Janela*), mas não necessariamente o inverso. As dependências são utilizadas sempre que quisermos indicar que algum item depende de outro. Graficamente, uma dependência é representada por linhas tracejadas, possivelmente com setas e ocasionalmente incluindo um rótulo, conforme mostra a Figura 12.

Figura 12 - Dependências

Segundo, uma *associação* é um relacionamento estrutural que descreve um conjunto de ligações, em que as ligações são conexões entre objetos. A agregação é um tipo especial de associação, representando um relacionamento estrutural entre o todo e suas partes. Graficamente, uma associação é representada por linhas sólidas, possivelmente direcionadas, ocasionalmente incluindo rótulos e, freqüentemente, contendo outros adornos, como nomes de papéis e multiplicidades, conforme mostra a Figura 13.

0..1 *

empregador funcionário

Figura 13 - Associações

Terceiro, uma *generalização* é um relacionamento de especialização/generalização, nos quais os objetos dos elementos especializados (os filhos) são substituíveis por objetos do elemento generalizado (os pais). Dessa maneira, os filhos compartilham a estrutura e o

comportamento dos pais. Graficamente, um relacionamento de generalização é representado como linha sólida com uma seta em branco apontando o pai, conforme mostra a Figura 14.

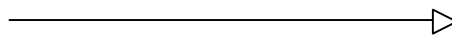



Figura 14 – Generalizações

Quarto, uma *realização* é um relacionamento semântico entre classificadores, em que um classificador especifica um contrato que outro classificador garante executar. Os relacionamentos de realizações serão encontrados em dois locais: entre interfaces e as classes ou componentes que as realizam; e entre casos de uso e as colaborações que os realizam. Graficamente, um relacionamento de realização é representado por uma linha tracejada com seta branca entre uma generalização e um relacionamento de dependência, conforme mostra a Figura 15.

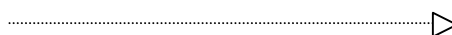



Figura 15 - Realização

Esses quatro elementos são os itens relacionais básicos que podem ser incluídos em um modelo de UML. Também existem variações desses quatro elementos, como refinamentos, rastros, inclusões e extensões (para dependências).

3.1.3. Diagramas na UML

Um *diagrama* é a apresentação gráfica de um conjunto de elementos, geralmente representadas como gráficos de vértices (itens) e arcos (relacionamentos). São projetados para permitir a visualização de um sistema sob diferentes perspectivas; nesse sentido, um diagrama constitui uma projeção de um determinado sistema.

Em todos os sistemas, com exceção dos mais triviais, um diagrama representa uma visão parcial dos elementos que compõem o sistema. O mesmo elemento pode aparecer em

todos os diagramas, em apenas alguns (o caso mais comum) ou em nenhum diagrama (um caso muito raro).

Na teoria, um diagrama pode conter qualquer combinação de itens e de relacionamentos. Na prática, porém, aparecerá um pequeno número de combinações comuns, que são consistentes com as cinco visões mais úteis da arquitetura de um sistema complexo de software. Por isso, a UML inclui nove desses diagramas: diagrama de classes, diagrama de objetos, diagrama de casos de uso, diagrama de seqüências, diagrama de colaborações, diagrama de gráficos de estados, diagrama de atividades, diagrama de componentes, diagrama de implantação.

Um *diagrama de classe* exibe conjunto de classes, interfaces e colaborações, bem como seus relacionamentos. Esses diagramas são encontrados com maior freqüência em sistemas de modelagem orientados a objeto e abrangem uma visão estática da estrutura do sistema. Os diagramas de classes que incluem classes ativas direcionam a perspectiva do processo estático do sistema.

Um *diagrama de objetos* exibe um conjunto de objetos e seus relacionamentos. Representa retratos estáticos de instâncias de itens encontrados em diagramas de classes. São diagramas que abrangem a visão estática da estrutura ou do processo de um sistema, como ocorre nos diagramas de classes, mas sob perspectiva de casos reais ou de protótipos.

Um *diagrama de caso de uso* exibe um conjunto de caso de uso e atores (um tipo especial de classe) e seus relacionamentos. Diagramas de caso de uso abrangem a visão estática de casos de uso do sistema. Esses diagramas são importantes principalmente para a organização e a modelagem de comportamentos do sistema.

Tanto os diagramas de seqüências como os de colaborações são tipos de diagramas de interações. Um *diagrama de interação* exibe uma interação, consistindo de um conjunto de

objetos e seus relacionamentos, incluindo as mensagens que podem ser trocadas entre eles. Diagramas de interações abrangem a visão dinâmica de um sistema.

Um *diagrama de seqüências* é um diagrama de interação cuja ênfase está na ordenação temporal das mensagens; o *diagrama de colaborações* é um diagrama de interação cuja ênfase está na organização estrutural dos objetos que enviam e recebem mensagens. Os diagramas de seqüências e de colaborações são isomórficos, o que significa que é possível transformar o diagrama de um tipo em um diagrama de outro tipo.

Os *diagramas de gráfico de estados* exibem uma máquina de estados, formada por estados, transições, eventos e atividades. Os diagramas de gráfico de estados abrangem a visão dinâmica de um sistema. São importantes principalmente para modelagem de comportamentos de uma interface, classe ou colaboração e para dar ênfase a comportamentos de um objeto ordenados por eventos, o que é de grande ajuda para modelagem de sistemas reativos.

Um *diagrama de atividade* é um tipo especial de diagrama de gráfico de estado, exibindo o fluxo de uma atividade para outra no sistema diagramas de atividades. Abrange a visão dinâmica do sistema e é importante principalmente para modelagem da função de um sistema e dá ênfase ao fluxo de controle entre objetos.

Um *diagrama de componente* exibe as organizações e as dependências existentes em um conjunto de componentes diagramas de componentes. Abrange a visão estática da implementação de um sistema. Está relacionado aos diagramas de classes, pois tipicamente os componentes são mapeados para uma ou mais classes, interfaces ou colaborações.

Um *diagrama de implantação* mostra a configuração dos nós de processamento em tempo de execução e os componentes neles existentes. Abrange a visão estática do funcionamento de uma arquitetura diagramas de implantação. Está relacionado aos diagramas de componentes, pois tipicamente um nó inclui um ou mais componentes.

3.2. Regras da UML

Os blocos de construção da UML não podem ser simplesmente combinados de uma forma aleatória. Como ocorre em qualquer linguagem, a UML tem um determinado número de regras que especificam o que deverá ser um modelo bem-formado. Os *modelos bem-formados* são aqueles auto-consistentes semanticamente e em harmonia com todos os modelos a eles relacionados.

As regras de formação nos fornecem o caminho para construirmos um modelo coerente, bem-formado, que respeite as regras de sintaxe e semântica a ele aplicado, permitindo uma correta interação entre os blocos de construção.

Toda linguagem possui sua *sintaxe* e *semântica*. Na UML, através da sintaxe, temos as regras que definem como os elementos da linguagem são dispostos dentro de expressões, e estas são combinadas. A semântica se refere ao significado dos elementos da linguagem. Ela define como as expressões sintáticas são associadas a um significado. Na UML, as regras semânticas especificam o que são e como podem ser aplicados seus diversos elementos. As regras sintáticas mostram notações e representações.

A UML dispõe de regras de formação que abrangem elementos como: nomes (quais nomes podem ser atribuídos a coisas, relacionamentos e diagramas), escopo (o contexto que determina um significado específico para um nome), visibilidade (como esses nomes podem ser vistos e utilizados pelos outros), integridade (como os itens se relacionam entre si de forma adequada e consistente), execução (o que significa executar ou simular um modelo dinâmico).

Os modelos construídos durante o desenvolvimento de sistemas complexos de software tendem a evoluir e podem ser visualizados pelos participantes de diferentes maneiras e em momentos distintos. Por isso, é comum a equipe de desenvolvimento não construir apenas

modelos bem-formados, mas também modelos: parciais (certos elementos ficam ocultos para simplificar a visão do modelo), incompletos (certos elementos podem ser omitidos), inconsistentes (a integridade do modelo não é assegurada).

É inevitável a utilização desses modelos que não são bem-formados como um detalhamento de um sistema em formação durante o ciclo de vida de desenvolvimento de um software. As regras da UML incentivam (mas não obrigam) a considerar as questões mais importantes de análise, projeto e implementação que levam esses modelos a se tornarem bem-formados ao longo do tempo.

3.3. Mecanismos básicos da UML

A UML se torna mais simples por causa da presença de quatro mecanismos básicos, aplicados de maneira consistente na linguagem: especificações, adornos, divisões comuns e mecanismos de extensão.

A UML é mais do que uma linguagem gráfica. Por trás de cada parte de suas notações gráficas, existe uma *especificação* capaz de fornecer uma declaração textual da sintaxe e da semântica do respectivo bloco de construção. A notação gráfica da UML serve para visualizar um sistema; a especificação da UML serve para determinar os detalhes do sistema. Levando em consideração esses dois aspectos, é possível construir modelos de maneira incremental, desenhando diagramas e depois acrescentando uma semântica às especificações do modelo ou diretamente pela criação de uma especificação, talvez com aplicação de engenharia reversa a um sistema existente, seguida pela criação dos diagramas que constituem em projeções nessas especificações.

As *especificações* da UML fornecem um repertório semântico, contendo todas as partes de todos os modelos de determinado sistema, cada parte relacionada às demais de uma forma consistente. Assim, os diagramas da UML são apenas projeções visuais a partir desse repertório, cada diagrama revelando um aspecto interessante específico do sistema.

Em sua maioria, os elementos da UML têm uma notação gráfica única e direta, que proporciona uma representação visual dos aspectos mais importantes do elemento. Por exemplo, a notação para uma classe é intencionalmente projetada para ser desenhada facilmente, pois as classes são os elementos mais comumente encontrados em sistemas de modelagem orientados a objetos. A notação de classe também expõe os aspectos mais importantes da classe, ou seja, seu nome, atributos e operações.

A especificação da classe pode incluir outros detalhes, como se a classe é abstrata ou como é a visibilidade de seus atributos e operações. Muitos desses detalhes podem ser representados como *adornos* gráficos ou textuais para anotação retangular básica da classe. Por exemplo, a Figura 16 mostra uma classe ornamentada com a finalidade de indicar que é uma classe abstrata com duas operações públicas, uma protegida e uma privada.

Todos os elementos da notação UML são iniciados com um símbolo básico, ao qual pode ser acrescentada uma variedade de adornos específicos desse símbolo.

Na modelagem de sistemas orientados a objetos, o mundo costuma ser dividido pelo menos de duas maneiras. Primeiro existe *a divisão de classes e objetos*. Uma classe é uma abstração; um objeto é uma manifestação concreta dessa abstração. Na UML, as classes podem ser modeladas, assim como os objetos, conforme mostra a Figura 17.

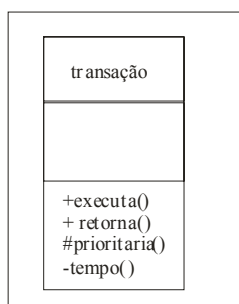



Figura 16 - Adornos

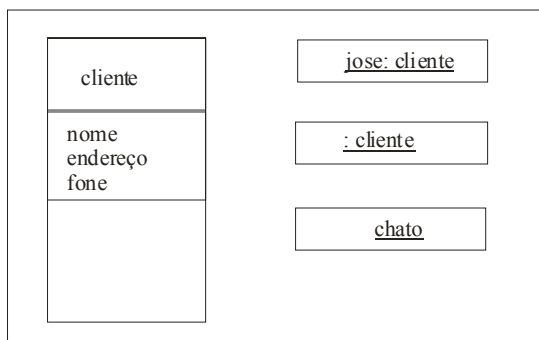



Figura 17 - Classes e objetos

Nessa figura, existe uma única classe, chamada *Cliente*, juntamente com três objetos: *jose* (marcado explicitamente como um objetos cliente); *:cliente* (um objeto cliente anônimo) e *chato* (em cuja especificação está marcado como sendo um tipo de objeto cliente, apesar de isso não ser apresentado aqui explicitamente).

Quase todos os blocos de construção disponíveis na UML apresentam o mesmo tipo de dicotomia de classe/objeto. Graficamente, para diferenciar os objetos, a UML utiliza os mesmos símbolos das respectivas classes e depois simplesmente sublinha os nomes dos objetos.

Segundo, existe *uma separação de interface e implementação*. Uma interface declara um contrato e a implementação representa uma realização completa desse contrato, responsável pela manutenção fiel da semântica completa da interface. Na UML, é possível modelar as interfaces e suas implementações, conforme mostra a Figura 18.

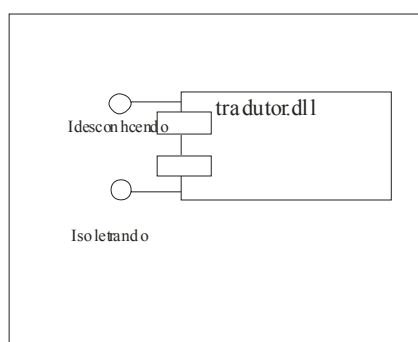



Figura 18 - Interfaces e suas implementações

Nessa figura, existe um componente chamado *tradutor.dll*, que implementa duas interfaces, *Idesconhecendo* e *Isoletrando.*

Quase todos os blocos de construção disponíveis na UML apresentam esse mesmo tipo de dicotomia interface/implementação.

A UML fornece uma linguagem-padrão para a elaboração de estrutura de projetos de software, mas não é possível que uma única linguagem fechada seja suficiente para expressar todas as nuances possíveis de todos os modelos em qualquer domínio o tempo todo. Por isso, a UML é aberta-fechada, permitindo que se possa ampliar a linguagem de uma maneira controlada. Os *mecanismos de extensibilidade* da UML incluem as seguintes características: estereótipos, valores atribuídos e restrições.

Um *estereótipo* amplia o vocabulário da UML, permitindo a criação de novos tipos de blocos de construção que são derivados dos já existentes, mas específicos a determinados problemas. Por exemplo, ao trabalhar com uma linguagem de programação, como Java ou C++, com freqüência há o desejo de modelar as exceções. Nessas linguagens, as exceções são apenas classes, apesar de serem tratadas de forma muito especial. Tipicamente, é suficiente permitir que sejam iniciadas e identificadas e nada mais. Em modelos, é possível transformar as exceções em cidadãos de primeira classe (isso significa que serão tratadas como blocos de construção básicos) marcando-as com um esterótipo adequado, como no caso da classe *Overflow* mostrada na Figura 19.

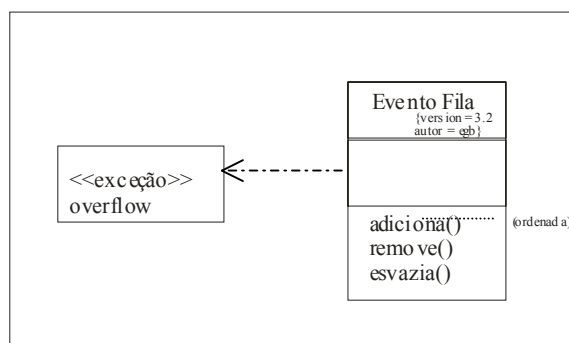



Figura 19 - Mecanismos de extensibilidade.

Um *valor atribuído* estende as propriedades dos blocos de construção da UML, permitindo a criação de novas informações na especificação de um elemento. Por exemplo, ao trabalhar em um produto que tenha muitas versões com o passar do tempo, certamente é desejável rastrear as versões e os autores de determinadas abstrações críticas. A versão e o autor não são conceitos primitivos da UML, mas podem ser acrescentados a qualquer bloco de construção, como uma classe, a partir da introdução de novos valores marcados a um certo bloco de construção. Na Figura 19, por exemplo, a classe *EventoFila* é estendida por uma marcação explícita de sua versão e autor.

Uma *restrição* amplia as semânticas dos blocos de construção da UML, permitindo acrescentar novas regras ou modificar as já existentes. Por exemplo, é possível restringir a classe *EventoFila* com a finalidade de que todos os acréscimos sejam feitos ordenadamente. Conforme mostra a Figura 19, é possível adicionar uma restrição e marcar explicitamente esses acréscimos para a operação *Adiciona*.

Em conjunto, esses três mecanismos de extensibilidade permitem definir a forma e ampliar a UML de acordo com as necessidades dos projetos. Esses mecanismos ainda permitem que a UML se adapte a novas tecnologias de software, como o possível surgimento de linguagens mais poderosas de programação distribuídas. É possível adicionar novos blocos de construção, modificar a especificação dos já existentes e até alterar sua semântica. Naturalmente, é importante fazer tudo isso de maneira controlada, para que, ao utilizar essas extensões o desenvolvedor permaneça fiel ao propósito da UML que é a comunicação de informações.

Conforme dito no capítulo anterior, seção 2.3.5, a UML é uma linguagem com as seguintes tarefas: visualizar; especificar; construir e documentar os artefatos de um sistema complexo de software. Essas tarefas requerem a visualização desses sistemas sob várias perspectivas, que chamamos de “arquitetura do sistema”, que será descrita na próxima seção.

3.4. Como a arquitetura de um sistema é expressada pela UML

Diferentes participantes (usuários finais, analistas, desenvolvedores, integradores de sistemas, o pessoal que testa os sistemas, escritores técnicos e gerentes de projetos), cada um traz contribuições próprias ao projeto e observa o sistema de maneira distinta em momentos diferentes ao longo do desenvolvimento do projeto. A arquitetura do sistema talvez seja o artefato mais importante a ser utilizado com o objetivo de gerenciar esses diferentes pontos de vista e, assim, tornar possível um controle do desenvolvimento iterativo e incremental de um sistema durante seu ciclo de vida (RUMBAUGH et al, 2000).

A arquitetura é o conjunto de decisões significativas acerca dos seguintes itens:

i. a organização do sistema de software;
ii. a seleção dos elementos estruturais e suas interfaces, que compõem o sistema;
iii. seu comportamento, conforme especificado nas colaborações entre esses elementos;
iv. a composição desses elementos estruturais e comportamentais em sub-sistemas progressivamente maiores;
v. o estilo de arquitetura que orienta a organização: os elementos estáticos e dinâmicos e suas respectivas interfaces, colaborações e composição.

A arquitetura de software não está apenas relacionada à estrutura e ao comportamento, mas também ao uso, à funcionalidade, ao desempenho, à flexibilidade, à reutilização, à abrangência, as adequações e as restrições de caráter econômico e tecnológico, além de questões estéticas.

Conforme mostra a Figura 20, a arquitetura de um sistema complexo de software pode ser descrita mais adequadamente por cinco visões interligadas. Cada visão constitui uma projeção na organização e estrutura do sistema, cujo foco está voltado para determinado aspecto desse sistema.

A *visão do caso de uso* abrange os casos de uso que descrevem o comportamento do sistema conforme é visto pelos usuários finais, analistas e pessoal de teste. Essa visão não especifica realmente a organização do sistema de um software. Porém, ela existe para especificar as forças que determinam a forma da arquitetura do sistema. Com a UML, os aspectos estáticos dessa visão são capturados em diagramas de caso de uso, enquanto os aspectos dinâmicos são capturados em diagramas de interação, diagramas de gráfico de estados e diagramas de atividades.

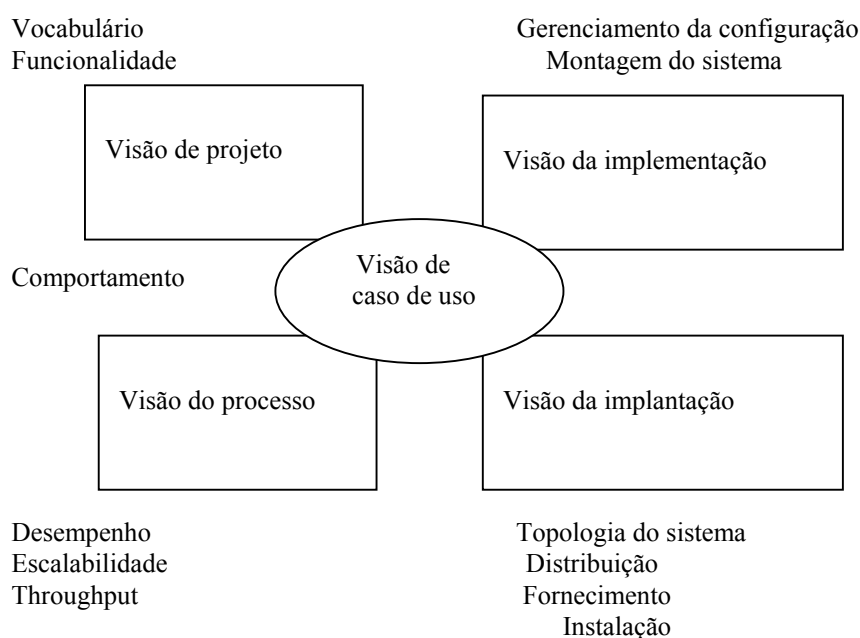



Figura 20 – a modelagem da arquitetura de um sistema

A *visão de projeto* de um sistema abrange as classes, interfaces e colaborações que formam o vocabulário do problema e de sua solução. Essa perspectiva proporciona principalmente um suporte para os requisitos funcionais do sistema, ou seja, os serviços que o sistema deverá fornecer a seus usuários finais. Com a UML, os aspectos estáticos dessa visão são captados em diagramas de classes e de objetos; os aspectos dinâmicos são captados em diagramas de interações, diagramas de gráfico de estados e diagramas de atividades.

A *visão do processo* abrange os threads e os processos que formam os mecanismos de concorrência e de sincronização do sistema. Essa visão cuida principalmente de questões

referentes ao desempenho, à escalabilidade e ao throughput do sistema. Com a UML, os aspectos estáticos e dinâmicos dessa visão são captados nos mesmos tipos de diagramas da visão de projeto, mas com o foco voltado para as classes ativas que representam esses threads e processos.

A *visão de implementação* de um sistema abrange os componentes e os arquivos utilizados para a montagem e fornecimento do sistema físico. Essa visão envolve principalmente o gerenciamento da configuração das versões do sistema, compostas por componentes e arquivos de alguma maneira independentes, que podem ser reunidos de diferentes formas para a produção de um sistema executável. Com a UML, os aspectos estáticos dessa visão são capturados em diagramas de componentes; os aspectos dinâmicos são capturados em diagramas de interações, de gráfico de estados e de atividades.

A *visão de implantação* de um sistema abrange os nós que formam a topologia de hardware em que o sistema é executado. Essa visão direciona principalmente a distribuição, o fornecimento e a instalação das partes que constituem o sistema físico. Com a UML, os aspectos estáticos dessa visão são capturados em diagramas de implantação; os aspectos dinâmicos são capturados em diagramas de interações, de gráfico de estados e diagramas de atividades.

Cada uma dessas cinco visões pode ser considerada isoladamente, permitindo que diferentes participantes dirijam seu foco para os aspectos da arquitetura do sistema que mais lhe interessam. Essas cinco visões também interagem entre si – os nós na visão de implantação contêm componentes da visão de implementação que, por sua vez, representa a realização física de classes, interfaces, colaborações e classes ativas provenientes das visões de projeto e de processo. A UML permite expressar cada uma dessas cinco visões e suas interações.

CONSIDERAÇÕES

A UML é apenas uma linguagem e, portanto, é somente uma parte de um método para desenvolvimento de software. Através de sua estrutura, conduz à criação e leitura de seus modelos, mas não determina *quais* e nem *quando* esses modelos precisam ser criados. Essa é uma responsabilidade do processo de desenvolvimento.

A UML é independente do processo, apesar de ser perfeitamente utilizada em processo orientado a casos de usos, centrado na arquitetura, iterativo e incremental. Cabe ao processo decidir quais artefatos serão produzidos, a equipe e atividades necessárias para criar e gerenciar esses artefatos.

No próximo capítulo vamos dar uma visão geral do que vem a ser o processo de desenvolvimento de software, e o porque de seu uso dentro de uma organização. Além disso, descreveremos algumas ferramentas utilizadas atualmente para captar e documentar as decisões tomadas, durante as fases do processo de desenvolvimento.

CAPÍTULO 4 – PROCESSOS DE DESENVOLVIMENTO DE SOFTWARE E FERRAMENTAS

Segundo Quatrani (2001) o segredo do sucesso para um projeto bem sucedido, é a adoção de uma notação, um processo e uma ferramenta. A notação ocupa uma parte importante em qualquer modelo, pois é ela que mantém o processo reunido.

Segundo Booch (1995), a notação tem três papéis:

i. serve como linguagem para comunicar às decisões que não são óbvias ou que não podem ser concluídas a partir do próprio código;
ii. oferece semânticas ricas o bastante para captar todas as importantes decisões estratégicas e táticas; e
iii. oferece uma forma concreta o bastante para os seres humanos racionalizarem e para as ferramentas manipularem.

Um projeto desenvolvido com sucesso satisfaz ou excede as expectativas do cliente, é desenvolvido em tempo e de maneira econômica, e é passível de mudança e adaptação. O ciclo de vida do desenvolvimento precisa promover criatividade e inovação. Ao mesmo tempo, o processo de desenvolvimento precisa ser controlado e medido, para garantir que o projeto seja, de fato, completado.

Quatrani (2001) afirma:

> A criatividade é essencial à habilidade de todas as arquiteturas baseadas em objeto bem estruturadas, mas os desenvolvedores que permitem criatividade completamente irrestrita tendem a nunca chegar a um fechamento. Da mesma forma, é necessária a disciplina ao organizar os esforços

> de uma equipe de desenvolvedores, mas muito mais disciplina dá uma origem a uma feia burocracia que mata todas as tentativas de inovação.

Qualquer método de desenvolvimento de software é melhor suportado com uma ferramenta. Atualmente, há muitas ferramentas no mercado – desde as simples ferramentas de projeto a sofisticadas ferramentas de modelagem de objetos.

Este capítulo se propõe a apresentar o porque do uso de um processo de produção de software que se adeque às necessidades da organização e de uma ferramenta para captar e documentar as decisões tomadas durante as fases do processo.

4.1.Processos

Um processo é um conjunto de passos parcialmente ordenados, constituídos por atividades, métodos, práticas e transformações, usado para atingir uma meta. Esta meta geralmente está associada a um ou mais resultados concretos finais, que são os produtos da execução do processo.

Um processo é uma receita a ser seguida por um projeto; o projeto concretiza uma abstração, que é o processo. Não se deve confundir um processo com o respectivo produto ou com a execução do processo através de um projeto.

Um processo é definido quando tem documentação que detalha: o que é feito (produto), quando (passos), por quem (agentes), as coisas que usa (insumos) e as coisas que produz (resultados). Processos podem ser definidos com mais ou menos detalhes, como acontece com qualquer receita. Os passos de um processo podem ter ordenação apenas parcial, o que pode permitir paralelismo entre alguns passos. Passos, agentes, insumos e resultados estão entre os elementos de um processo. A arquitetura de um processo define um arcabouço conceitual para a organização dos elementos de um processo. Uma discussão aprofundada sobre definição de processos pode ser encontrada em (HUMPHREY, 1995).

Segundo Larman (2000) “um processo de desenvolvimento de software é um método para organizar as atividades relacionadas com a criação, entrega e manutenção de sistemas de software”. Na subseção a seguir forneceremos uma visão muito breve às atividades fundamentais de um processo de desenvolvimento de software.

4.1.1.Processos de Desenvolvimento de Software

Em Engenharia de Software, processos podem ser definidos para atividades como desenvolvimento, manutenção, aquisição e contratação de software. Pode-se também definir subprocessos para cada um destes; por exemplo, um processo de desenvolvimento abrange subprocessos de determinação dos requisitos, análise, projeto, implementação e testes. Em um processo de desenvolvimento de software, o ponto de partida para a arquitetura de um processo é a escolha de um modelo de ciclo de vida.

A seguir descreveremos sucintamente alguns processos de desenvolvimento de software para ilustrar o que foi dito, são eles:

i. Processo Pessoal de Software de Humphrey (1995);
ii. Processo de Software para Times de Humphrey (1999);
iii. Processo Orientado a Objetos para Software extensível de Filho (1998);
iv. Processo Unificado de Rumbaugh, Booch, e Jacobson (2000).

4.1.1.1.O Processo Pessoal de Software

Watts Humphrey (1995) propôs uma série de processos pessoais que pudessem ser aprendidos em uma disciplina de engenharia de software. Esse conjunto é chamado de Processo Pessoal de Software (Personal Software Process), ou PSP.

Esses processos são aprendidos através de uma seqüência de pequenos projetos. Os projetos devem ser realizados seguindo rigorosamente os processos, que incluem um conjunto de formulários, scripts e relatórios predefinidos. Os projetos são individuais, com duração típica de cerca de 10 horas.

O PSP possui uma série de estágios, onde em cada estágio são introduzidos elementos novos de processo. Estes estágios também possuem uma classificação. As classificações existentes são: processos pessoais básicos, processos pessoais com planejamento, processos pessoais com gestão da qualidade, processos pessoais cíclicos. Ver Tabela 2.

Classificação	**Nome**	**Elementos novos de processo**
Processos pessoais básicos	PSP0	Registro de tempos Registro de defeitos Padronização dos tipos de defeitos
	PSP0.1	Padronização da codificação Medição do tamanho Proposição de melhorias de processo
Processos pessoais com planejamento	PSP1	Estimativas de tamanho Relatório de testes
	PSP1.1	Planejamento de tarefas Planejamento de cronogramas
Processos pessoais com gestão da qualidade	PSP2	Revisões de código Revisões de projeto
	PSP2.1	Modelos de projeto
Processos pessoais cíclicos	PSP3	Desenvolvimento cíclico

Tabela 2 – Os estágios do PSP.

A Tabela 3 apresenta detalhes do PSP3, versão final do processo, que inclui os elementos introduzidos em todos os processos anteriores. O PSP3 tem um ciclo de vida de entrega em estágios. Não existe um tratamento separado dos requisitos; estes são muito simples em todos os projetos, e as respectivas atividades são consideradas parte do planejamento. O planejamento inclui a estimativa de tamanhos (medidos em linhas de código, com base em um modelo conceitual orientado a objetos), de esforços (medidos em tempo de desenvolvimento), de cronograma (tempo físico) e de defeitos.

Fase	Atividades	Resultados
Planejamento	Especificação de requisitos. Estimativa de tamanho. Estratégia. Estimativa de recursos. Estimativa de prazos. Estimativa de defeitos.	Documentos dos requisitos. Modelo conceitual. Planos de recursos, prazos e qualidade. Registro de tempos.
Projeto de alto nível	Especificações externas. Projeto dos módulos. Prototipagem. Estratégia de desenvolvimento. Documentação da estratégia de desenvolvimento. Registro de acompanhamento de problema.	Especificações funcionais. Especificações de estados. Roteiros operacionais. Especificações de reutilização. Estratégia de desenvolvimento. Estratégia de testes. Registro de tempos.
Revisão do projeto de alto nível.	Verificação da cobertura do projeto. Verificação da máquina de estados. Verificação lógica. Verificação da consistência do projeto. Verificação da reutilização. Verificação da estratégia de desenvolvimento. Conserto de defeitos.	Projeto de alto nível revisto. Estratégia de desenvolvimento revista. Estratégia de testes revista. Registro de defeitos de projeto de alto nível. Registro de problemas de projeto de alto nível. Registro de tempos.
Desenvolvimento	Projeto do módulo. Revisão do projeto. Codificação. Revisão do código. Compilação. Teste. Reavaliação e reciclagem.	Projeto detalhado dos módulos. Código dos módulos. Registro de defeitos dos módulos. Registro de problemas dos módulos. Relatório dos testes. Registro de tempos.
Post-mortem	Contagem de defeitos injetados e removidos. Contagem de tamanhos e tempos.	Resumo do projeto.

Tabela 3 – Detalhes do PSP

O projeto é feito de acordo com padrões rigorosos, que usam conceitos de orientação a objetos, síntese lógica e máquinas seqüenciais, submetido a uma fase rigorosa de verificação. Com base no projeto, a fase de desenvolvimento é dividida em ciclos; cada ciclo inclui

projeto detalhado, codificação revisão do código, compilação e testes de unidade dos respectivos módulos. Ao final de cada ciclo, o planejamento é reavaliado.

O PSP sempre termina com uma fase *post-mortem*, na qual é feito um balanço final do projeto. As lições aprendidas são documentadas e analisadas, para melhoria do processo do projeto seguinte.

4.1.1.2.O Processo de Software para Times

Como seqüência natural do PSP, Humphrey (1999) introduziu o Processo de Software para Times (Team Software Process), ou TSP.

Fase	**Atividades**
Lançamento	Descrição do curso: visão geral; informação para os alunos; objetivos do produto. Formação dos times: integrantes, metas e reuniões. Primeira reunião do time: requisitos de dados. Ativação dos projetos.
Estratégia	Visão geral da estratégia de desenvolvimento. Critérios da estratégia de desenvolvimento. Seleção da estratégia de desenvolvimento. Documentação da estratégia de desenvolvimento. Estimativas de tamanho. Definição do processo de controle de mudanças.
Planejamento	Visão geral do plano de desenvolvimento. Produção do plano de tarefas. Produção do cronograma. Produção dos planos pessoais dos engenheiros. Balanceamento de carga dos engenheiros. Produção do plano de qualidade.
Requisitos	Revisão do processo de requisitos. Revisão das demandas dos usuários. Esclarecimento das demandas dos usuários. Distribuição das tarefas de requisitos. Documentação dos requisitos. Revisão dos requisitos. Colocação dos requisitos na linha de base. Revisão dos requisitos pelos usuários.

Tabela 4 – Fases do TSPe – parte 1

A Tabela 4 e a Tabela 5 apresentam as partes principais da versão publicada deste processo (TSPe), que é orientada para utilização educacional. O TSP usa um modelo em

espiral; os passos mostrados nessas tabelas correspondem a um dos ciclos. Ao longo de 15 semanas, são executados tipicamente três ciclos de desenvolvimento de um produto.

Fase	**Atividades**
Projeto	Revisão do processo de projeto. Projeto de alto nível. Distribuição das tarefas de projeto. Documentação do projeto. Revisão do projeto. Atualização do projeto, com colocação da linha de base.
Implementação	Revisão do processo de implementação. Distribuição das tarefas de implementação. Projeto detalhado. Inspeção do projeto detalhado. Código. Inspeção do código. Teste de unidades. Revisão da qualidade dos componentes. Liberação dos componentes.
Testes	Revisão do processo de testes. Planejamento e desenvolvimento dos testes. Construção. Integração. Testes de sistema. Documentação dos testes.
Post-mortem	Revisão do processo de post-mortem. Revisão dos dados de processo. Avaliação do desempenho dos papéis. Preparação do relatório do ciclo. Revisão dos pares.

Tabela 5 – Fases do TSPe – parte 2

Os participantes do time de desenvolvedores são organizados de tal forma que cada desenvolvedor desempenhe um ou dois papéis gerenciais bem definidos, além de dividir a carga de desenvolvimento. Os papéis suportados pelo processo são os de gerente de desenvolvimento, de planejamento, de qualidade, de processo e de suporte, além do líder do time.

O planejamento e o controle rigoroso de tamanhos, esforços, prazos e defeitos, característicos do PSP, continuam a ser feitos. O TSP enfatiza algumas áreas que

correspondem às áreas do nível 2 do CMM: gestão dos requisitos, planejamento e controle de projetos, garantia da qualidade e gestão de configurações.

4.1.1.3.O Processo Orientado a Objetos para Software Extensível

O Processo Orientado a Objetos para Software Extensível (PROSE) foi desenvolvido dentro do Departamento de Ciência da Computação da Universidade Federal de Minas Gerais para uso em projetos de desenvolvimento de sistemas de apoio à Engenharia de Telecomunicações (FILHO, 1998).

O PROSE foi concebido com um processo padrão que visava a cobrir todo o ciclo de vida dos produtos de software, especialmente de aplicativos extensíveis, através de sucessivas versões produzidas durante um ciclo de vida de produto com duração de vários anos. Esses produtos normalmente seriam aplicativos gráficos interativos, baseados na tecnologia orientada a objetos.

O processo completo inclui um conjunto de recomendações, padrões, roteiros de revisão, políticas e modelos. A última versão publicada desse processo está contida em (FILHO, 1998a), (FILHO, 1998b), (FILHO, 1998c), (FILHO, 1998d). Uma característica central do PROSE é o uso da tecnologia orientada a objetos nas atividades de análise, projeto e implementação. Os modelos produzidos no processo usam a notação UML.

A Tabela 6 descreve a estrutura básica do PROSE. O modelo de ciclo de vida de cada projeto é o da entrega evolutiva. O ciclo de vida de um produto é constituído por versões sucessivas, que são produzidas em diferentes projetos; portanto, o modelo de ciclo de vida de produto é em espiral.

Macroatividade	**Fase**	**Subfase**
Ativação		
Especificação	Engenharia dos Requisitos	Levantamento dos requisitos
		Detalhamento dos requisitos

	Planejamento	Planejamento do desenvolvimento.
		Planejamento da qualidade
Desenvolvimento	Projeto	Projeto dos testes de aceitação.
		Projeto arquitetônico.
		Projeto das interfaces de usuário.
		Planejamento das Liberações executáveis.
	Implementação	Construção da liberação executável 1
		Construção da liberação executável 2
		Construção da liberação executável ...
		Preparação da implantação
		Testes alfa
Implantação	Testes beta	
	Operação piloto	

Tabela 6 – Atividades do PROSE

4.1.1.4.O Processo Unificado

Rumbaugh, Booch, e Jacobson (2000) propuseram a UML como uma notação de modelagem orientada a objetos, independente de processos de desenvolvimento. Além disso, propuseram o Processo Unificado (Unified Process), que utiliza a UML como notação de uma série de modelos que compõem os principais resultados das atividades do processo.

O Processo Unificado descende de métodos anteriores propostos pelos autores em (BOOCH, 1994), (BOOCH, 1996), (JACOBSON, 1994b), (JACOBSON, I.; ERICSSON, M.; JACOBSON, A., 1994a), (JACOBSON, 1997), (JACOBSON, 1999) e (RUMBAUGH, 1991).

Segundo seus autores, o Processo Unificado apresenta as seguintes características centrais: é dirigido por casos de uso; é centrado na arquitetura; é iterativo e incremental.

As atividades de desenvolvimento são *orientadas por casos de uso*. O Processo Unificado atribui uma forte ênfase à elaboração de sistemas com base em uma compreensão completa a respeito de como o sistema entregue será utilizado. As noções de casos de uso e de cenários são empregados para alinhar o fluxo do processo a partir da captura dos requisitos por meio de testes e para proporcionar threads que poderão ser acompanhados ao longo do desenvolvimento até o sistema entregue.

Sob a orientação do Processo Unificado, o desenvolvimento é *centrado na arquitetura*. O processo focaliza o desenvolvimento inicial e a linha de base da arquitetura de um software. Dispor de uma arquitetura robusta facilita o desenvolvimento paralelo, minimiza a necessidade de refazer o trabalho e aumenta a probabilidade de reutilização de componentes e a capacidade de manutenção eventual do sistema. Esse projeto de arquitetura serve como uma base sólida para o planejamento e desenvolvimento de software a partir de componentes.

O Processo Unificado é um processo *iterativo*. No caso de sistemas simples, poderá ser perfeitamente viável definir seqüencialmente todo o problema, estruturar a solução inteira, elaborar o software e então testar o produto final. Entretanto, devido à complexidade e sofisticação exigidas pelos sistemas atuais, esse procedimento linear para o desenvolvimento de sistemas não é realista. Uma solução iterativa requer uma compreensão crescente do problema por meio de aperfeiçoamentos sucessivos e do *desenvolvimento incremental* de uma solução efetiva em vários ciclos. Inerente à solução iterativa é a flexibilidade para a acomodação de novos requisitos ou de mudanças táticas de objetivos de negócio. Também permite que o projeto identifique e solucione riscos de início, em vez de posteriormente.

Segundo Quatrani (2001) "o Processo Unificado é estruturado junto a duas dimensões: tempo (divisão do ciclo de vida em fases e iterações) e componentes de processo (produção de um conjunto específico de artefatos com atividades bem definidas). Ambas as dimensões precisam ser levadas em conta para um projeto ser bem sucedido".

O ciclo de vida de um produto tem um modelo espiral, em que cada projeto constitui um ciclo, que entrega uma liberação do produto. O Processo Unificado não trata do que acontece entre ciclos. Cada ciclo é dividido nas fases mostradas na Tabela 7. Uma fase é o período de tempo entre dois importantes marcos de progresso do processo em que um conjunto bem-definido de objetivos é alcançado, artefatos são concluídos e decisões são tomadas em relação à passagem para a fase seguinte.

Fase	Descrição
Concepção	Fase na qual se justifica a execução de um projeto de desenvolvimento de software, do ponto de vista do negócio do cliente.
Elaboração	Fase na qual o produto é detalhado o suficiente para permitir um planejamento acurado da fase de construção.
Construção	Fase na qual é produzida uma versão completamente operacional do produto.
Transição	Fase na qual o produto é colocado à disposição de uma comunidade de usuários.

Tabela 7 – Fases do Processo Unificado

Uma característica importante do Processo Unificado é que as atividades técnicas são divididas em subprocessos chamados de *fluxos de trabalho* (workflows), mostrado na Tabela 8. Cada fluxo de trabalho (que chamaremos simplesmente de fluxo) tem um tema técnico específico, enquanto as fases constituem divisões gerenciais, caracterizadas por atingirem metas bem definidas.

Fluxo	Descrição
Requisitos	Fluxo que visa a obter um conjunto de requisitos de um produto, acordado entre cliente e fornecedor.
Análise	Fluxo cujo objetivo é detalhar, estruturar e validar os requisitos, de forma que estes possam ser usados como base para o planejamento detalhado.
Projeto	Fluxo cujo objetivo é formular um modelo estrutural do produto que sirva de base para a implementação.
Implementação	Fluxo cujo objetivo é realizar o projeto em termos de componentes de código
Testes	Fluxo cujo objetivo é verificar os resultados da implementação.

Tabela 8 – Fluxos do Processo Unificado.

Um produto comercial baseado no Processo Unificado é o Rational Unified Process (RUP); ele contém uma base de conhecimento que detalha e estende o material apresentado em (JACOBSON, 1999).

4.1.1.4.1.Rational Unified Process - RUP

O Rational Unified Process dispõe de suporte para *técnicas orientadas a objetos* (Rumbaugh; Booch; Jacobson, 2000). Cada modelo é orientado a objetos. Os modelos do RUP são baseados nos conceitos de objetos e classes e nos relacionamentos existentes entre eles e utilizam a UML como sua notação comum.

Segundo Rumbaugh, Booch e Jacobson (2000):

> As atividades do RUP dão ênfase à criação e manutenção de *modelos* no lugar de documentos impressos. O caráter racional subjacente ao foco em modelos no lugar de documentos impressos, empregado pelo RUP, consiste em minimizar a sobrecarga associada à geração e manutenção de documentos e maximizar o conteúdo das informações relevantes.

O RUP encoraja o *controle de qualidade* e o *gerenciamento de riscos,* contínuos e objetivos. A avaliação da qualidade é inserida no processo, em todas as atividades envolvendo todos os participantes, com a utilização de medidas e critérios objetivos. Não é tratada como algo pensado tardiamente ou como uma atividade separada. O gerenciamento de riscos é inserido no processo, de forma que os riscos para o sucesso do projeto são identificados e atacados no início do processo de desenvolvimento, quando há tempo suficiente para uma reação adequada.

A passagem pelas quatro principais fases é chamada um ciclo de desenvolvimento e resulta na geração de um software. O primeiro passo das quatro fases é chamado o ciclo inicial de desenvolvimento. A menos que a vida do produto seja interrompida, um produto existente evoluirá para sua próxima geração pela repetição da mesma seqüência de fases de concepção, elaboração, construção e transição. Isso é a evolução do sistema, de modo que os ciclos de desenvolvimento posteriores aos ciclos iniciais são seus ciclos de evolução.

O RUP é composto por nove fluxos de trabalho de processo, são eles:

i. modelagem de negócio – descreve a estrutura e a dinâmica da empresa;
ii. requisitos – descreve o método baseado em casos de uso para identificar requisitos;

iii. análise e projeto – descreve as várias visões da arquitetura;

iv. implementação – leva em consideração o desenvolvimento do software, o teste da unidade e a integração;

v. teste – descreve casos de teste, procedimentos e medidas para acompanhamento de erros;

vi. entrega – abrange a configuração do sistema a ser entregue;

vii. gerenciamento da configuração – controla as modificações e mantém a integridade dos artefatos do projeto;

viii. gerenciamento de projeto – descreve várias estratégias para o trabalho com um processo iterativo;

ix. Ambiente – abrange a infra-estrutura necessária para o desenvolvimento do sistema.

Capturado em cada fluxo de trabalho de processo está um conjunto de atividades e artefatos correlacionados. Um *artefato* é algum documento, relatório ou executável, que é produzido, manipulado ou consumido. Uma *atividade* descreve as tarefas – criando, realizando e verificando etapas – executadas pelos trabalhadores para criar ou modificar artefatos, juntamente com as técnicas e diretrizes para a realização de tarefas, possivelmente incluindo a utilização de ferramentas para ajudar a automação de algumas das tarefas.

Conexões importantes entre os artefatos estão associadas a alguns desses fluxos de trabalho de processo. Por exemplo, o modelo de caso de uso gerado durante a captação de requisitos *é realizado pelo* modelo de implementação a partir do processo de implementação e *verificado pelo* modelo de teste a partir do processo de teste.

Cada atividade do RUP tem artefatos associados, ou exigidos como uma entrada ou gerados como uma saída. Alguns artefatos são utilizados com a finalidade de direcionar a

entrada para atividades subseqüentes, mantidos como recursos de referência sobre o projeto ou gerados em um formato como entregas contratuais.

Os modelos são o tipo mais importante de artefato do RUP. São nove modelos que, em conjunto, abrangem todas as decisões importantes para a visualização, especificação, construção e documentação de um sistema complexo de software:

i. modelo de negócio – estabelece uma abstração da empresa;
ii. modelo de domínio – estabelece o contexto do sistema;
iii. modelo de caso de uso – estabelece os requisitos funcionais do sistema;
iv. modelo de análise (opcional) – estabelece o projeto de uma idéia;
v. modelo de projeto – estabelece o vocabulário do problema e de sua solução;
vi. modelo de processo (opcional) – estabelece os mecanismos de concorrência e de sincronização do sistema;
vii. modelo de implantação – estabelece a topologia do hardware em que o sistema é executado;
viii. modelo de implementação – estabelece as partes utilizadas para montar e liberar o sistema físico;
ix. modelo de teste – estabelece os caminhos pelos quais o sistema é validado e verificado.

Uma visão é uma projeção em um modelo. No RUP, a arquitetura de um sistema é capturada em cinco visões interligadas: a visão de projeto, a visão de processo, a visão de implantação, a visão de implementação e a visão de caso de uso.

Qualquer método de desenvolvimento é melhor suportado com uma ferramenta. A seguir trataremos de algumas ferramentas existente no mercado para modelagem utilizando UML.

4.2.Ferramentas

Atualmente, há muitas ferramentas no mercado – tudo, desde as simples ferramentas de projeto a sofisticadas ferramentas de modelagem de objeto. A proposta desta seção é apresentar algumas funcionalidades de algumas ferramentas existente no mercado que fazem uso da notação UML. Dentre as ferramentas pesquisadas estão:

i. Rational Rose 7.5;

ii. System Architect 9.0;

iii. Poseidon for UML Community Edition.

4.2.1.A Ferramenta Rational Rose versão 7.5

A família de produtos Rational Rose foi projetada para oferecer ao desenvolvedor de software um conjunto completo de ferramentas de modelagem visual para o desenvolvimento de soluções estáveis, eficientes, atendendo as reais necessidades nos ambientes cliente/servidor e sistemas de tempo real. Os produtos Rational Rose compartilham um padrão universal comum, tornando a modelagem acessível e mais fácil (QUATRANI, 2001).

A ferramenta de modelagem da Rational Rose foi projetada para ser utilizada por todos os membros da equipe de desenvolvimento de software, com a finalidade de construir software de uma maneira melhor e mais rápida, desde o início do processo de desenvolvimento. É uma ferramenta completa, robusta e que possui flexibilidade em todos os ambientes de modelagem (desenvolvimento web, modelagem de dados com Java, Visual Studio, e C++). Essa ferramenta permite aos desenvolvedores, gerentes de projetos, engenheiros e analistas visualizarem, entenderem e refinarem os requisitos e arquitetura antes de partir para a codificação, eliminando esforços inúteis no ciclo de desenvolvimento. É uma ferramenta simples de modelagem para ser usada durante todo o ciclo de desenvolvimento do software, assegurando a escalabilidade e a confiabilidade do sistema e permitindo o

atendimento das necessidades do cliente. Se há a necessidade de cuidar das mudanças de requisitos, acompanhar cada fase de iteração do desenvolvimento, estar atento as diferentes tecnologias usadas no projeto, ou manter um único nível de comunicação, esta ferramenta faz tudo isso com rapidez e simplicidade (RATIONAL, 2003).

Dentre os benefícios que a ferramenta proporciona, estão:

i. a comunicação, através de uma linguagem única, entre todos os membros da equipe para gerenciar um projeto complexo de desenvolvimento;

ii. otimização dos esforços dos analistas, projetistas, desenvolvedores e engenheiros, reduzindo o tempo da entrega do produto;

iii. é uma ferramenta completa, com suporte mundial para treinamento, simplificando a necessidade tecnológica e maximizando o investimento.

A ferramenta de modelagem da Rational Rose possibilita uma melhor comunicação, uma equipe de desenvolvimento reduzida, maior facilidade para entender os sistemas, e uma melhor visão da sua estrutura complexa de software.

Os gerentes de projeto, analistas, projetistas e desenvolvedores falam diferentes linguagens e trabalham em diferentes ambientes. Essencialmente, os problemas acontecem por causa da pobre comunicação entre os membros da equipe. Para obter sucesso no desenvolvimento de software, todos os membros da equipe de desenvolvimento devem se comunicar através de uma linguagem única. O acesso ao dado e a estrutura da aplicação são altamente interdependentes, mas no passado estes grupos trabalhavam relativamente isolados devido a carência de uma solução integrada.

A ferramenta de modelagem da Rational Rose permite também que a equipe de desenvolvimento trabalhe junto com completa integração entre diferentes arquivos de modelos para o completo controle de versão. Se na equipe de desenvolvimento existem pessoas que usam outras ferramentas da Rational, a ferramenta de modelagem tem

funcionalidade embutida e total integração com a solução completa do ciclo de vida da Rational.

Usando um ambiente de modelagem em conjunto com o ambiente de desenvolvimento que usa a linguagem UML, o Rational Rose e a sua solução de controle de configuração habilita todos os membros da equipe a usar uma ferramenta e uma linguagem para comunicação eficiente. Operações individuais em áreas de trabalho privadas contêm uma visão pessoal do modelo inteiro, com a habilidade para modificar aquele ou aquela parte do modelo até que esteja pronto para ser compartilhado.

Uma ferramenta é um benefício, somente, se puder ser adaptável as necessidades dos usuários e seja fácil de usar. A ferramenta de modelagem da Rational Rose e a UML aproximam os analistas, projetistas e desenvolvedores usando somente uma ferramenta e uma linguagem, mesmo que eles estejam usando diferentes versões da ferramenta.

O desenvolvimento de software é complexo. A ferramenta de modelagem da Rational Rose usa uma linguagem única o que pode significar a diferença entre sucesso e falha. 74% de todos os projetos de software falham por causa da excessiva complexidade, constantes mudanças de requisitos, objetivos obscuros e carência nas informações dos usuários (RATIONAL, 2003).

Os modelos criados com a ferramenta de modelagem da Rational Rose, são um bom ponto de partida para desenvolvedores e engenheiros para entender o escopo completo do projeto, fazer a engenharia-reversa dos códigos existentes para o modelo, e a análise/recomendações das características do sistema e iterações.

Uma arquitetura aberta provê flexibilidade com relação a mudanças. Permite uma divisão clara de trabalho entre os membros da equipe, reduzindo ambigüidades e melhorando a comunicação entre todos. Isso também minimiza a aflição quando novos requisitos são adicionados no ciclo de vida do projeto/produto ou quando membros da equipe são movidos

para outros projetos, permitindo uma suave integração, menor confusão e melhor organização (RATIONAL, 2003).

Uma arquitetura baseada em componente habilita a contrução de código reutilizável e escalável e a construção de diferentes componentes em conjunto. Com a habilidade para ver e usar a interface separadamente da lógica do negócio e do dado, a solução da Rational permite que os processos de software evoluam controladamente, gerenciadamente e melhor identificados. Se essas evoluções envolvem mudanças nas linguagens ou plataformas, ou há necessidade de adaptação de tecnologias futuras, a solução da Rational Rose ajuda o gerenciamento das mudanças no desenvolvimento, permitindo que a equipe de desenvolvimento seja capaz de reusar componentes existentes, reduzindo o tempo de lançamento no mercado, e obtendo vantagem estratégica necessária para se estar na liderança do mercado.

Integração com sistemas legados, a necessidade de testes freqüentes de procedimentos, e suporte a múltiplos ambientes de desenvolvimento integrado - IDEs é justamente um pouco do dia-a-dia das mudanças encontradas no desenvolvimento de projetos. A Rational tem soluções poderosas para estes problemas. Incorpora componentes existentes e novos rapidamente e integra facilmente os modelos gerados a partir da engenharia reversa a UML. O Rational Rose é a única solução que oferece uma capacidade de desenvolvimento *round*-trip – muda o código, o modelo ou ambos, e facilmente sincroniza-os a qualquer hora. Evita o conflito de testes dos componentes e o risco da migração pela automação do processo diretamente do modelo UML; não somente melhorará a eficiência e a facilidade do seu processo de teste inteiro, mas reduzirá também o risco, melhorando a previsibilidade, ganhando tempo e dinheiro, e evitando conflitos. O Rational Rose trabalha com qualquer IDE e com as últimas tecnologias, então ela poder ser a ferramenta de desenvolvimento única para todas as necessidades (RATIONAL, 2003).

4.2.2.A Ferramenta Poseidon for UML versão Community Edition

O Poseidon for UML Community Edition (BOGER, 2003) é uma versão gratuita do Poseidon for UML que traz algumas vantagens entre elas: instalação facilitada, interface melhorada e suporte a todos os diagramas da UML. É bom lembrar que o Poseidon em todas as suas versões utiliza a engine do Argo UML incrementando a ferramenta free com funcionalidades extras.

Por ser escrito em Java, o Poseidon suporta todas as plataformas para as quais exista uma máquina virtual Java. Os modelos criados com a ferramenta são ponto de partida para desenvolvedores fazerem a engenharia-reversa dos códigos existentes em Java para o modelo assim como a geração de código a partir dos modelos.

A instalação no Windows é bastante simples e pode ser feita de três maneiras: usando o assistente de instalação, descompactando o pacote .ZIP ou através do Java Web Start, que executa o poseidon a partir do site dos desenvolvedores na Internet. Além do Windows o Poseidon trabalha com uma série de outros sistemas operacionais como: Linux, Solaris, Aix, HP, Ux, Irix, Mac-OS.

Estão disponíveis todos os nove diagramas da UML obedecendo as especificações da versão 1.3. Nas versões pagas, os diagramas podem ser importados de arquivos .mdl, do Rational Rose. A versão gratuita não conta com essa opção. A impressão dos modelos não está habilitada nessa versão gratuita. Esses problemas podem ser contornados exportando para os formatos GIF, PNG, JPG, Postscript (PS), Encapsulated Postscript (EPS), Scalable Vector Graphics (SVG) and Windows Meta File (WMF).

Uma funcionalidade interessante do Poseidon é o painel de críticas. Esse mecanismo varre o modelo em busca de incoerências.

O Poseidon possui uma série de plug-ins que introduzem novas funcionalidades.

4.2.3.System Architect versão 9.0

O System Architect (HARMON, 2003) se propõe a ser a única ferramenta para toda a modelagem de negócios da empresa, dividida pelo fabricante em três etapas: identificação dos processos atuais, redefinição dos processos e automatização, através de sistemas informatizados, de todos os processos possíveis. Seguindo essa proposta, o System Architect excede muito as expectativas de quem busca apenas um ferramenta para modelagem de sistemas em UML. Isso, em muitos casos, faz com que o Analista acabe pagando por recursos que nunca irá usar.

O System Architect contém um amplo conjunto de modelos inter-relacionados e matrizes de informação permitindo que o analista venha fazer a modelagem de toda a organização, a partir de várias perspectivas diferentes. Dispõe ainda de uma ferramenta que permite a utilização de modelos previamente elaborados por diversas empresas como templates para o desenvolvimento de novos projetos. Possui suporte completo para a UML 1.4. Também pode-se utilizar outras técnicas já estabelecidas, como a OMT (RUMBAUGH et. al, 1994), Booch (1994) (1995), Shlaer/Mellor (1988) (1994), Coad/Yourdon (1992) (1993)e Jacobson (1994b).

O System Architect mantém os documentos produzidos na análise de requisitos diretamente no repositório, de onde podem ser visualizados, modificados ou mesmo utilizados em outros projetos.

Realiza engenharia reversa bi-direcional, tanto para C++ e Java, além de gerar código para várias outras linguagens, incluíndo CORBA IDL, VB, Delphi e Powerbuilder - Tudo em uma mesma ferramenta. Também possui engenharia reversa e geração de tabelas, visões e trigers para as principais sistemas gerenciadores de banco de dados relacional, incluindo

Oracle, Sybase, DB2, SQL Server e AS/400. Gera modelos de entidade-relacionamento, com a separação entre modelos lógico e físico.

Possui também acesso bi-direcional para o PowerBuilder e geração de código para Visual Basic, Delphi, ou PowerBuilder a partir de informações contidas no modelo de dados.

O System Architect é ferramenta líder no segmento da análise estruturada; o analista pode escolher entre várias técnicas: Gane and Sarson (1982), Yourdon (1978) (1992), DeMarco (1979), Ward and Mellor (1985), Engenharia da Informação (Information Engineering), ou mesmo a técnicas de análise e projeto estruturados de sistemas (Strutured Systems Analysis and Design Methodology - SSADM). Também possibilita a criação de diagramas de decomposição automaticamente, a partir de diagramas de fluxo de dados. Através da customização de seu repositório, o System Architect pode inclusive ser utilizado para implementar Análise Essencial, ou acomodar-se a metodologia em uso na sua empresa.

O System Architect é a única ferramenta pronta para atender a necessidade de migração da modelagem estruturada para orientação a objeto.

Com relação a relatórios e geração de documentação em HTML o System Architect possui um pacote de relatórios e documentação contém: um sistema interno baseado em SQL que permite a criação de mais de 150 relatórios fornecidos com o produto, ou a criação de novos com uma interface de criação gráfica; um link com o Word para a utilização de Templates; um gerador de HTML com a criação automática de hyperlinks e relatórios.

Com relação ao *design de interface gráfica* o System Architect permite que o analista visualize a interface de seu sistema com o usuário. Pode-se criar protótipos de telas gráficas e menus para aplicativos Windows, bem como telas de caracter para aplicativos em Cobol. As telas podem ser automaticamente criadas através do "drag-and-drop" de objetos definidos na enciclopédia do StarTeam, podendo então ter estas telas implementadas através dos arquivos .FRM do Visual Basic, ou .DLG para Windows e arquivos .MNU.

CONSIDERAÇÕES

Neste capítulo mostramos os benefícios de se utilizar um processo e uma ferramenta no desenvolvimento de software. Dentre os quais podemos destacar: tempo de entrega dentro dos prazos, maior economia, melhor controle das atividades realizadas, facilidade de adaptação do software depois de implantado, facilidade de comunicação entre os componentes da equipe de desenvolvimento, etc.

Descrevemos também um produto comercial que é o Rational Unified Process (RUP) com o objetivo de conhecê-lo, para que possamos escolher o processo que será utilizado pela equipe de informática da Fundação HEMOAM no desenvolvimento do sistema para doação de sangue do interior no estado do Amazonas. No capítulo 6 (estudo de caso), deste trabalho, descreveremos o processo que será utilizado.

A tabela 9 mostra um resumo das ferramentas pesquisadas. A ferramenta escolhida para auxiliar no desenvolvimento do sistema citado é a Rational Rose 7.5. A escolha deu-se devido à mesma ser a mais completa, e adequar-se a todo o ciclo de desenvolvimento.

No próximo capítulo vamos abordar tecnologias para comunicação de dados, com o objetivo de conhecê-las, pois a idéia do trabalho é modelar um sistema de informação para o interior do estado, que troque dados com o sistema atual existente na capital (Manaus).

Ferramenta	**Resumo**
Rational Rose versão 7.5	• suporta todos os diagramas da UML; • pode ser utilizada por todos os membros da equipe de desenvolvimento, desde o início do processo de desenvolvimento, mantém um único nível de

	comunicação; • completa e possui flexibilidade em todos os ambientes de modelagem; • possibilita gerenciamento de mudanças de requisitos, acompanhamento de cada fase do desenvolvimento; • permitir a visualização, entendimento e o refinamento do software antes da codificação; •ideal para quem pensa em usar a linguagem Java, pois facilita o trabalho de programação.
Poseidon for UML versão Community Edition	• suporta todos os diagramas da UML; • impressão dos modelos não habilitada nesta versão. Possibilita a exportação dos modelos para outros formatos; • realiza engenharia reversa bi-direcional para a linguagem Java.
System Architect versão 9.0	• suporta diagramas da UML, de todas as principais técnicas orientadas a objetos e da técnica estruturada; • pode ser usada para o gerenciamento de processos; • realiza engenharia reversa bi-direcional para as linguagens C++ e Java.

Tabela 9 - Resumo das ferramentas pesquisadas

CAPÍTULO 5 – TECNOLOGIAS PARA COMUNICAÇÃO DE DADOS

As organizações estão se tornando cada vez mais *organizações conectadas em redes*. A Internet e as redes do tipo Internet dentro da organização (intranets), entre uma organização e seus parceiros comerciais (extranets) e outros tipos de redes se tornaram a principal infra-estrutura de informática de muitas organizações. Essas redes de telecomunicações permitem a gerentes, usuários finais, equipes e grupos de trabalho trocarem eletronicamente dados e informações em qualquer parte do mundo com outros usuários, clientes, fornecedores e parceiros de negócios. As empresas e grupos de trabalho podem assim colaborar de modo mais criativo; gerenciar suas operações e recursos de modo mais eficaz; e, competir com sucesso na atual economia globalizada em rápida transformação.

Segundo O'brien (2002):

> *Telecomunicações* é toda forma de troca de informações (por exemplo, voz, dados, texto e imagens) por meio de redes computadorizadas. Em geral, uma *rede de telecomunicações* é qualquer arranjo no qual um *emissor* transmite uma mensagem para um *receptor* por um *canal* que consiste em algum tipo de *veículo*.

Na próxima seção vamos descrever como os computadores estão fisicamente conectados, ou seja, os tipos de redes de telecomunicações.

5.1.Tipos de Redes de Telecomunicações

Os computadores localizados em escritórios de empresas estão fisicamente conectados a redes locais (Local Área Networks - LANs), que estão localizadas dentro de uma área geográfica limitada como um andar, departamento ou edifício. As redes locais conectam computadores, impressoras, scanners e dispositivos compartilhados como modems, unidades

de videoconferência e unidades de fax. As redes locais são conectadas a outras redes locais configurando redes metropolitanas (Metropolitan Area Networks – MANs) e redes geograficamente distribuídas (Wide Área Networks – WANs) (DODD, 2000).

A diferença entre redes locais, redes metropolitanas, e redes geograficamente distribuídas é a distância com que os dispositivos podem comunicar-se entre si. Descreveremos a seguir, cada uma delas.

5.1.1.Redes Locais (LANs)

Segundo O'brien (2002):

> As *redes locais* (LANs) conectam computadores e outros dispositivos de processamento de informações dentro de uma área física limitada, como um escritório, sala de aula, um prédio, uma fábrica ou outro estabelecimento de trabalho. As LANs têm se tornado corriqueiras em muitas organizações por proporcionarem capacidades de rede de telecomunicações que conectam usuários finais em escritórios, departamentos e outros grupos de trabalho.
> As LANs utilizam vários meios principais de telecomunicações, tais como cabeamento telefônico comum, cabo coaxial ou até sistemas de rádio sem fio, para interconectarem estações de trabalho de microcomputadores e periféricos de computador. Para se comunicar com a rede, cada PC normalmente dispõe de uma placa de circuito chamada *placa de interface de rede*. A maioria das LANs utilizam um micro computador mais potente que dispõe de um disco rígido de grande capacidade, chamado *servidor de arquivo* ou *servidor de rede*, que contém um programa de *sistema operacional de rede* que controla as telecomunicações e o uso e compartilhamento dos recursos da rede.

Antes de descrevermos as redes metropolitanas, vamos falar brevemente das *redes cliente/servidor* que se tornaram a arquitetura predominante de informações na computação nas empresas. Vale ressaltar que esta é a arquitetura predominante de rede na sede da Fundação HEMOAM (Manaus).

5.1.1.1.Redes Cliente/Servidor

A maioria das LANs utiliza a tecnologia *cliente/servidor*. Em uma *rede cliente/servidor*, o PC do usuário final ou estações de trabalho são os clientes. Eles são interconectados por redes locais e compartilham o processamento de aplicações com servidores de rede, que também gerenciam as redes. As redes locais também são interconectadas com outras LANs e redes remotas de estações de trabalho clientes e servidores.

Segundo O’brien (2002):

> Uma tendência em curso é a redução (downsizing) de sistemas maiores pela sua substituição por redes cliente/servidor. Uma rede cliente/servidor de várias redes locais interconectadas, por exemplo, pode substituir uma grande rede baseada em mainframe por muitos terminais de usuários finais. Isto, normalmente, envolve um esforço complexo e dispendioso para instalar os novos softwares aplicativos que substituirão os softwares de sistema de informação mais velhos e tradicionais (mainframe), agora chamados de *sistemas legados*. As redes cliente/servidor são vistas como mais econômicas e flexíveis dos que os sistemas legados no atendimento de necessidades do usuário final, grupo de trabalho e unidade de negócios e mais adaptáveis para se ajustarem a uma gama diversificada de cargas de trabalho computacional.

Na seção 5.5 estenderemos os conhecimentos sobre arquitetura de informação de redes e descreveremos também os protocolos de rede.

5.1.2.Redes Metropolitanas (MANs)

Segundo Tanenbaum (1996) “uma rede metropolitana, ou MAN, é, na verdade, uma versão ampliada de uma LAN”. Uma MAN pode abranger um grupo de escritórios vizinhos ou uma cidade inteira e pode ser privada ou pública.

As *redes metropolitanas*, ou MANs, são conexões entre redes locais, que ocorrem dentro de uma cidade. Como exemplos de MANs podemos citar grandes hospitais e complexos universitários. Ao invés de transportar registros e radiografias entre duas localizações, o hospital aluga linhas telefônicas de alta capacidade para transmissão dos registros e imagens. As conexões entre esses locais são conexões de MAN. Essas conexões podem ser alugadas de uma companhia telefônica ou construídas pela organização (DODD, 2000).

5.1.3.Redes Geograficamente Distribuídas (WANs)

Segundo Tanenbaum (1996) “uma rede geograficamente distribuída, ou WAN, abrange uma ampla área geográfica, com freqüência um país ou um continente”.

As redes de telecomunicações que cobrem uma ampla área geográfica são chamadas de *redes geograficamente distribuídas*. Essas grandes redes se tornaram uma necessidade para realizar as atividades cotidianas de muitas empresas e organizações governamentais e de seus

usuários finais. As WANs, por exemplo, são utilizadas por muitas empresas multinacionais para transmitir e receber informações entre seus funcionários, clientes, fornecedores e outras organizações ao longo de cidades, regiões, países e o mundo (O'BRIEN, 2002).

A variedade de conexões de rede geograficamente distribuída disponíveis é complexa. A seleção de uma conexão apropriada de rede geograficamente distribuída depende da quantidade de tráfego entre localizações, da qualidade do serviço necessário, do preço e da compatibilidade com os sistemas de computador localizados dentro das organizações (DODD, 2002).

Na próxima seção descreveremos a rede de telefonia pública que é o principal veículo para transportar tráfego de voz, fax e modem discado.

5.2. A rede pública

Embora novas redes estejam sendo construídas para tratar de dados de alta velocidade, voz e serviços baseados na Internet de maneira mais eficiente, a *comutação de circuitos* ainda é o principal veículo para transportar tráfego de voz, fax e modem discado.

A International Telecommunications Union (ITU) define comutação como "o estabelecimento por demanda de uma conexão individual a partir de uma entrada desejada para uma saída desejada dentro de um conjunto de entradas e saídas pelo tempo necessário para transferir as informações" (DODD, 2000).

O objetivo desta seção é explicar as diferenças entre serviços dedicados e serviços comutados. A principal diferença entre esses serviços é que as chamadas comutadas são discáveis. As conexões feitas discando um número de telefone são flexíveis, isto é, variam conforme o número de telefone discado. Com os serviços dedicados os enlaces são permanentes.

Podemos citar como exemplos de serviços comutados as linhas telefônicas residenciais e comerciais. Os serviços comutados têm as seguintes características:

i. são discados, ou seja, os usuários discam um número de telefone para criar uma conexão temporária;
ii. são tarifados de acordo com o uso, encontrando-se as tarifas baseadas na quantidade de tempo em que as chamadas mantêm uma conexão;
i. são utilizados para tráfego de voz, vídeo, imagem e dados;
ii. são utilizados com linhas telefônicas analógicas e digitais;
iii. são liberados (a linha telefônica e o equipamento na rede) quando os chamadores desligam, para serem utilizados por outra pessoa ou dispositivo de dados.

Serviços dedicados, também chamados linhas privadas, são mais especializados que as linhas discadas. As organizações as utilizam para economizar dinheiro quando precisam transmitir grandes quantidades de dados ou estabelecer chamadas telefônicas de voz com determinados locais. As organizações têm o uso da linha telefônica 24 horas um dia, sete dias por semana. Elas pagam uma taxa mensal simples pelo uso da linha. Não há taxas extras independentemente de quanto a linha é utilizada. Serviços de linha dedicada ou privada têm as seguintes características:

i. linhas privadas, dedicadas têm uma configuração fixa;
ii. usuários pagam uma taxa mensal simples, não há tarifas por minuto;
iii. voz, vídeo, imagem e dados são utilizados;
iv. linhas telefônicas analógicas e digitais são utilizadas;
v. uso exclusivo das pessoas ou dos dispositivos conectados a linha dedicada.

A seguir vamos descrever os componentes básicos de uma rede de telecomunicações.

5.3.Componentes básicos de uma rede de telecomunicações

Uma rede de telecomunicações consiste em cinco categorias básicas de componentes:

i. terminais;

ii. processadores de telecomunicações;

iii. canais de telecomunicações;

iv. computadores;

v. software de controle de telecomunicações.

Seja qual for o tamanho e complexidade que as redes de telecomunicações pareçam ter na realidade, essas cinco categorias básicas de componentes de rede devem estar funcionando para apoiar as atividades de telecomunicações de uma organização (O’BRIEN, 2002). A seguir descreveremos cada uma delas.

5.3.1.Terminais

Terminais, tais como computadores pessoais interconectados, redes de computadores ou terminais de vídeo. Todo dispositivo de entrada/saída que utiliza redes de telecomunicações para transmitir ou receber dados é um terminal, inclusive os telefones e os terminais de computador (O’BRIEN, 2002).

5.3.2.Processadores de telecomunicações

Processadores de telecomunicações apóiam a transmissão e recepção de dados entre terminais e computadores. Eles convertem, por exemplo, dados digitais em analógicos e vice-versa, codificam e decodificam dados e controlam a velocidade, precisão e eficiência do fluxo de comunicações entre computadores e terminais em uma rede de telecomunicações (O’BRIEN, 2002).

Os processadores de telecomunicações executam uma série de funções de controle e suporte em uma rede de telecomunicações. Podemos citar:

i. modems;

ii. multiplexadores;

iii. processadores de internetwork.

5.3.2.1.Modems

Os *modems* são o tipo mais comum de processador de comunicações. Eles convertem os sinais digitais de um computador ou terminal de transmissão em uma das extremidades de uma conexão de comunicações em freqüências analógicas que possam ser transmitidas por linhas telefônicas comuns. Um modem na outra extremidade da linha de comunicações converte os dados transmitidos de volta para a forma digital no terminal receptor (Oppenheimer, 1999).

5.3.2.2.Multiplexadores

Um *multiplexador* é um processador de comunicações que permite que um canal de comunicações isolado veicule transmissões simultâneas de dados de diversos terminais. Dessa forma, uma única linha de comunicações pode ser compartilhada por vários terminais.

5.3.2.3.Processadores de Internetwork

As redes de telecomunicações são interconectadas por processadores de comunicações com finalidades especiais chamados *processadores de internetwork*, como:

i. chave;

ii. roteador;

iii. hub;

iv. gateway.

Uma *chave* é um processador de comunicações que faz conexões entre os circuitos de telecomunicações em uma rede para que uma mensagem de telecomunicações possa alcançar seu destino pretendido.

Um *roteador* é um processador de telecomunicações mais inteligente que interconecta redes baseadas em diferentes regras ou *protocolos* para que uma mensagem de telecomunicações possa ser encaminhada até seu destino.

Um *hub* é um processador de comunicações para comutação de portas. Versões avançadas de hubs fornecem comutação automática entre conexões chamadas *portas* para acesso compartilhado aos recursos de uma rede.

Um *gateway* é um processador de comunicações que interconecta redes que utilizam diferentes arquiteturas de comunicações.

Todos estes dispositivos são essenciais para fornecer conectividade e fácil acesso entre as múltiplas LANs e redes remotas que fazem parte das intranets e redes cliente/servidor em muitas organizações.

5.3.3.Canais de telecomunicações

Canais de telecomunicações pelos quais os dados são transmitidos e recebidos. Os canais de telecomunicações utilizam combinações de *mídias de telecomunicações* para interconectar os demais componentes de uma rede de telecomunicações (O'BRIEN, 2002).

Entre as *mídias de telecomunicações* se incluem:

i. fio de pares trançados;

ii. cabos coaxiais;

iii. cabos de fibra ótica;

iv. microonda terrestre;

v. satélites de comunicações;

vi. sistemas de telefonia celular;

vii. LAN sem fio (wireless).

5.3.3.1.Fio de Pares Trançados

O cabo telefônico comum, que consiste de fio de cobre trançado em pares (*fio de pares trançados*), é o meio mais utilizado para telecomunicações. Essas linhas são utilizadas em redes estabelecidas de comunicações em todo o mundo tanto para transmissão de voz como de dados. Dessa forma, a fiação de pares trançados é utilizada amplamente em sistemas telefônicos domésticos e comerciais e em muitas redes locais e remotas.

5.3.3.2.Cabo Coaxial

O *cabo coaxial* consiste em um fio rígido de cobre ou alumínio envolto em espaçadores para seu isolamento e proteção. O revestimento e isolamento do cabo minimiza a interferência e a distorção dos sinais que o cabo conduz. Essas linhas de alta qualidade podem ser instaladas sob o chão e estendidas nos leitos dos lagos e oceanos. Permitem a transmissão de dados em alta velocidade e são usadas em lugar de linhas de fios de pares trançados em áreas metropolitanas com grande volume de serviço, para sistemas de TV a cabo e para conexão de curta distância entre computadores e equipamentos periféricos. Os cabos coaxiais também são utilizados para redes locais em muitos prédios comerciais e outros estabelecimentos de trabalho.

5.3.3.3.Fibra Ótica

As *fibras óticas* utilizam cabos constituídos de um ou mais filamentos capilares de fibra de vidro envolvidos em uma capa protetora. Podem conduzir pulsos luminosos cerca de 60 vezes mais do que o cabo coaxial e 3.000 vezes melhor do que as linhas de fios de pares trançados.

Os cabos de fibra ótica não são afetados por radiação eletromagnética e não geram esta radiação; por isso, múltiplas fibras podem ser colocadas no mesmo cabo. Os cabos de fibra

ótica possuem necessidade mínima de repetidoras para retransmissões de sinais, ao contrário dos meios principais de telecomunicações em cabo elétrico.

5.3.3.4.Microonda Terrestre

A *microonda terrestre* diz respeito a sistemas de microondas por terra que transmitem sinais de rádio de alta velocidade em um caminho de linha de mira entre estações repetidoras espaçadas a uma distância de aproximadamente 50 quilômetros. As antenas de microondas são normalmente instaladas no topo dos edifícios, torres, colinas e picos de montanhas, e já fazem parte da paisagem em muitas áreas do país. Ainda são um meio popular tanto para redes de longa distância como de áreas metropolitanas.

5.3.3.5.Satélites de Comunicações

Os *satélites de comunicações* também utilizam microonda de rádio como mídia de telecomunicações. Os satélites são movidos por painéis e podem transmitir sinais de microondas a uma taxa de várias centenas de milhões de bits por segundo. Eles servem como estações repetidoras para sinais de comunicações transmitidos de estações terrestres. As estações de terra utilizam pequenas antenas parabólicas para irradiar sinais de microondas para os satélites que amplificam e retransmitem os sinais para outras estações terrestres a milhares de quilômetros de distância.

Embora os satélites de comunicações fossem inicialmente utilizados para transmissão de voz e vídeo, estão sendo agora usados também para transmissão em alta velocidade de grande volume de dados.

5.3.3.6.Sistemas de Telefonia Celular

Os *sistemas de telefonia celular* utilizam várias tecnologias de comunicações por rádio. Entretanto, todas elas dividem uma área geográfica em pequenas áreas, ou *células*, normalmente uma área de um a vários quilômetros quadrados. Cada célula tem seu próprio dispositivo transmissor de baixa potência ou antena repetidora de rádio para retransmitir chamadas de uma célula para outra. Computadores e outros processadores de comunicações coordenam e controlam as transmissões de usuários de telefone móvel em sua passagem de uma área para outra.

5.3.3.7.Redes Locais LANs sem Fio (Wireless)

Cabear um escritório ou prédio para uma rede local muitas vezes é uma tarefa difícil e dispendiosa. Prédios mais velhos não possuem conduítes para cabos coaxiais ou fios de pares trançados, e os conduítes nos prédios mais novos podem não ter espaço suficiente para passar mais cabos adicionais. Uma solução para esses problemas é instalar uma *LAN sem fio*, utilizando uma dentre várias tecnologias sem fio (LAN por rádio; LAN infravermelha).

Uma LAN por rádio usa uma tecnologia de rádio de alta freqüência similar ao celular digital, ou uma tecnologia de rádio de baixa freqüência chamada *spread spectrum* (espectro de difusão).

A outra tecnologia LAN infravermelha utiliza raios de luz infravermelha para estabelecer conexões de rede entre computadores LAN.

5.3.4.Computadores

Computadores de todos os tamanhos e tipos são interconectados pelas redes de telecomunicações para que possam realizar suas tarefas de processamento de informações. Um mainframe, por exemplo, pode funcionar como um *computador anfitrião* ou *principal* (host) para uma grande rede, auxiliado por um computador de médio porte que funciona como

um *processador de front-end*, enquanto um microcomputador pode atuar como um *servidor de rede* para uma pequena rede de estações de trabalho de microcomputadores (O'BRIEN, 2002).

5.3.5.Software de controle de telecomunicações

Software de controle de telecomunicações consiste em programas que controlam atividades de telecomunicações e gerenciam as funções das redes de telecomunicações. Entre os exemplos se incluem todos os tipos de programas de gerenciamento, tais como *monitores de telecomunicações* para computadores principais, *sistemas operacionais de rede* para servidores de rede de microcomputadores e *navegadores de rede* para microcomputadores (O'BRIEN, 2002).

Muitos fabricantes de software oferecem software de telecomunicações conhecidos como *middleware*[4] (sistema operacional, programa de controle de rede local, e sistema de gerenciamento de banco de dados DBMS[5]), que podem ajudar diversas redes a se comunicarem entre si.

Os programas de gerenciamento de redes determinam prioridades de transmissão, encaminham (comutam mensagens), consultam terminais na rede e formam linhas de espera (filas) de pedidos de transmissão. Eles também detectam e corrigem erros de transmissão, registram estatísticas de atividade de rede e protegem recursos da rede contra acesso não autorizado.

Depois de entender os componentes básicos de uma rede de telecomunicações, entre eles os canais de telecomunicações, vamos, na próxima seção, descrever as alternativas de largura de banda que um canal de telecomunicações pode ter para transmitir seus dados.

[4] Software que fica entre o programa aplicativo e o programa de controle.
[5] DataBase Management System

5.4.Alternativas de Largura de Banda

Segundo O'brien (2002) "a velocidade e capacidade de comunicações das redes de telecomunicações podem ser classificadas por *largura de banda"*.

A largura de banda é a faixa de freqüência de um canal de telecomunicações; ela determina a taxa máxima de transmissão do canal. A velocidade e capacidade das taxas de transmissão de dados são normalmente medidas em bits por segundo (BPS). Esta medida é, às vezes, chamada de *baud* (taxa da velocidade de transmissão), embora *baud* seja, mais propriamente, uma medida de mudanças de sinal em uma linha de transmissão.

Os *canais analógicos de baixa velocidade* (banda de voz) são normalmente utilizados para taxas de transmissão de 300 a 9.600 BPS, mas podem agora suportar até 1 milhão de BPS (MBPS). Costumam ser linhas de fios de pares trançados não blindados, geralmente utilizadas para comunicações de dados por microcomputadores, terminais de vídeo e máquinas de fax.

Os *canais de velocidade média* (banda média) utilizam linhas de fios de pares trançados blindados para velocidades de transmissão de 9.600 BPS até 100 MBPS.

Os *canais digitais de alta velocidade* (banda larga) permitem taxas de transmissão a intervalos específicos de 256.000 BPS a vários bilhões de BPS. Normalmente utilizam transmissão por microonda, fibra ótica ou satélite. Ver Tabela 8 (O'BRIEN, 2002).

Tipo de mídia	BPS máximo
Pares trançados – não blindados/blindados	2M – 100M
Cabo coaxial – banda-base/banda larga	264M – 550M
Satélite/microonda terrestre	100M
LAN por Rádio sem fio	3.3M
LAN por infravermelho	4M
Cabo de fibra ótica	30G

KBPS = milhares de BPS ou kilobits por segundo
MBPS = milhões de BPS ou megabits por segundo

GBPS = bilhões de BPS ou gigabits por segundo.

Tabela 10 – Exemplos das velocidades de transmissão de telecomunicações por tipo de mídia.

Na próxima seção descreveremos brevemente os conceitos de *arquitetura* e *protocolo de rede*.

5.5.Arquitetura e Protocolos de Rede

Segundo O'brien (2002) "um *protocolo* é um conjunto padrão de regras e procedimentos para o controle das comunicações em uma rede. Entretanto, esses padrões podem estar limitados a apenas um equipamento de fabricante, ou apenas um tipo de comunicações de dados".

Até muito recentemente, havia uma falta de padrões suficientes para as interfaces entre o hardware, o software e os canais de comunicações das redes de comunicações de dados. Essa situação atrapalhava o uso de comunicações de dados, aumentava seus custos e reduzia sua eficiência e eficácia. Por esse motivo, é muito comum encontrar uma falta de compatibilidade entre o hardware e o software de comunicações de dados de diferentes fabricantes. Em resposta, os fabricantes de computadores e organizações nacionais e internacionais têm desenvolvido padrões chamados *protocolos* e planos mestres chamados *arquiteturas de redes* para apoiar o desenvolvimento de redes avançadas de comunicações de dados.

Parte do objetivo das *arquiteturas de redes* de comunicações é criar mais padronização e compatibilidade entre os protocolos de comunicações. Um exemplo de protocolo é um padrão para as características físicas dos cabos e conectores entre terminais, computadores, modems e linhas de comunicações. Outros exemplos são os protocolos que estabelecem as informações de controle de comunicações necessárias para o *handshaking* (aperto de mãos), que é o processo de trocar sinais e caracteres predeterminados para estabelecer uma sessão de

telecomunicações entre terminais e computadores. Outros protocolos lidam com o controle da recepção da transmissão de dados em uma rede, técnicas de comutação, conexões com a rede e assim por diante.

A meta das *arquiteturas de rede* é promover um ambiente aberto, simples, flexível e eficiente de telecomunicações. Isto é feito mediante o uso de protocolos padrão, hardware e software padrão para interfaces e o desenho de uma interface padrão de vários níveis entre os usuários finais e os sistemas de computadores.

Na próxima seção vamos conhecer as topologias de rede, ou estruturas, nas redes de telecomunicações.

5.6.Topologias de Rede

A topologia de uma rede de telecomunicação refere-se à forma de como os meios de transmissão e os nós de comutação estão organizados, determinando os caminhos físicos existentes e utilizáveis entre quaisquer pares de estações conectadas a essa rede (SOARES; LEMOS e COLCHER, 1995).

Segundo O'brien (2002) "existem diversos tipos básicos de *topologias* de rede. Três topologias básicas são utilizadas nas redes de telecomunicações locais e remotas". São elas:

i. estrela, que liga computadores de usuários finais a um computador central;
ii. anel, que liga processadores de computadores locais em um anel em uma base mais uniforme;
iii. barramento, que é uma rede na qual os processadores locais compartilham o mesmo barramento ou canal de comunicações.

Cada rede de computadores pode ser classificada em algumas poucas categorias básicas, dependendo de sua topologia. Uma topologia de barramento consiste em um cabo único, compartilhado, ao qual, muitos computadores se ligam. Quando usa um barramento, um

computador transmite um sinal que todos os outros computadores acoplados ao barramento recebem. Uma topologia em anel consiste em computadores conectados em um loop fechado. O primeiro computador está conectado ao segundo, o segundo conectado ao terceiro, e assim por diante, até o último computador, que está conectado ao primeiro. Finalmente, uma topologia em estrela se assemelha a uma roda, com a rede propriamente dita correspondendo a um hub central, e os links para computadores individuais correspondendo aos aros (COMER, 2001).

O'brien (2002) faz uma comparação entre as topologias de rede:

> As redes estrela, anel e barramento diferem em desempenho, confiabilidade e custo. Uma rede estrela pura é considerada menos confiável do que uma rede anel, já que os outros computadores na estrela são muito dependentes do computador central principal. Se ele falhar, não há processamento de reserva e capacidade de comunicações e os computadores locais são desligados uns dos outros. Portanto, é essencial que o computador principal seja altamente confiável. Dispor de algum tipo de arquitetura de multiprocessador para fornecer uma capacidade tolerante a falhas é uma solução comum.
> As redes anel e barramento são as mais comuns nas redes locais. As redes anel são consideradas mais confiáveis e menos dispendiosas para o tipo de comunicações nessas redes. Se um computador no anel falhar, os outros computadores poderão continuar a processar seu próprio trabalho, além de se comunicarem entre si.

Segundo Comer (2001):

> Cada topologia tem vantagens e desvantagens. Uma topologia em anel torna mais fácil aos computadores coordenarem o acesso e detectarem se a rede está operando corretamente. Porém, uma rede inteira em anel é desativada se um dos cabos é cortado. Uma topologia em estrela ajuda a proteger a rede de danos em um único cabo porque cada cabo conecta somente uma máquina. Um barramento exige menos fios que uma estrela, mas tem a mesma desvantagem de um anel: uma rede é desativada se alguém acidentalmente corta o cabo principal.

Na seção seguinte vamos mostrar os serviços existentes de redes digitais de alta velocidade, como funcionam, quais as aplicações e onde se enquadram.

5.7.Serviços de rede especializados

Segundo Dodd (2000) alguns dos serviços de rede especializados são:

i. Integrated Services Digital Network (ISDN);

ii. Digital Subscriber Line (DSL);

iii. Frame Relay;

iv. Asynchronous Transfer Mode (ATM);

v. Synchronous Optical Network (SONET).

A Rede Digital de Serviços Integrados (ISDN) é um serviço de transporte de dados digitais oferecido por operadoras telefônicas regionais. Admite transmissão de texto, imagens gráficas, vídeo, música, voz e outros materiais de origem sobre linhas telefônicas. Oferece uma solução de acesso remoto econômica para pessoas que trabalham em casa e escritórios remotos que exigem velocidades de transmissão mais altas e estabelecimento de conexões mais rápidas que os links dial-up analógicos podem oferecer. A ISDN também é uma boa escolha como um link de backup para outro tipo de link; por exemplo, um link de Frame Relay. O custo de um circuito ISDN normalmente se baseia em uma taxa mensal mais o tempo de uso (OPPENHEIMER, 1999).

A Linha Digital de Assinante (DSL) é outra tecnologia que está recebendo atenção no caso de acesso remoto. As empresas de telefonia oferecem esta linha para tráfego de dados de alta velocidade sobre fios de telefônicos comuns. Com a DSL, um escritório doméstico ou um pequeno escritório pode conectar um modem DSL ou outro dispositivo a uma linha telefônica e usar essa conexão para alcançar uma intranet de site central. Além disso, alguns provedores de serviços da Internet (ISPs) fazem acordos com as empresas de telefonia que permitem aos provedores oferecerem a seus clientes acesso à Internet através de um modem DSL.

A DSL é semelhante à ISDN pelo fato de ser uma tecnologia que opera sobre linhas telefônicas existentes entre uma estação de comutação telefônica e uma casa ou escritório. Porém, a DSL usa esquemas sofisticados de modulação para oferecer velocidades muito mais altas que a ISDN (OPPENHEIMER, 1999).

À medida que se acelerou a necessidade de largura de banda para WANs, as companhias telefônicas atualizaram suas redes internas para usar as tecnologias SONET (Rede Óptica Síncrona) e ATM (Modo de Transferência Assíncrono), e começaram a

oferecer novos serviços a seus clientes, como SMDS (Switched Multimegabit Data Service – Serviço de Dados Comutado de Vários Megabits) e Frame Relay.

O Frame Relay é um serviço de rede de dados com valor agregado utilizado para conectar redes locais distantes entre si para acesso à Internet e acesso remoto. Cada uma das localizações de uma organização que deseja utilizar o serviço de frame relay tem uma linha com a rede de frame relay da concessionária. As localizações não precisam manter linhas e dispositivos de comunicações de dados para conexões com cada localização. A concessionária se responsabiliza pelo planejamento da capacidade e confiabilidade de rede (DODD, 2000).

O ATM permite que as concessionárias e organizações transportem uma mistura de voz, vídeo e comunicações de dados com diferentes qualidades de serviço. O ATM é utilizado principalmente em redes de Internet, redes celulares, redes de concessionárias e redes de frame relay. As comutações de ATM transportam voz, dados, imagem e vídeo em velocidades muito altas de até 13,22 gigabits sobre cabeamento de fibra ótica. O ATM permite utilizar uma tecnologia de comutação para voz, vídeo e dados em sua rede de backbones onde altos volumes de tráfego são concentrados (DODD, 2000).

Serviço de rede	**Locais onde geralmente é utilizado**	**Como é utilizado**
ISDN 2 canais de voz ou dados + 1 canal de sinalização ou 23 canais de voz/dados + 1 canal de sinalização	Clientes residenciais, organizações. Provedores de serviço da Internet, distribuidores automáticos de chamadas.	Videoconferência, telecomutação, acesso à Internet. Centrais de chamada, acesso remoto a bancos de dados corporativos
DSL 128 Kbps a 6 megabits	Telecomutadores, corporações, provedores de serviços da Internet, cliente residencial.	Acesso remoto a bancos de dados corporativos e acesso à Internet; alguns tipos utilizados para voz.
Frame Relay 56 Kbps a 45 megabits	Mídia para clientes comerciais grandes	Um público, principalmente serviço de rede de dados para rede local para conexões de rede local e para acesso

		remoto.
ATM 56 Kbps para 2,5 gigabits	Companhias telefônicas, provedores de serviços da Internet, redes de frame relay e organizações grandes como universidades importantes.	Utilizado para comutação de voz em backbone de alto volume, vídeo e tráfego de dados.
SONET Até 129.000 canais em cabo de fibra ótica	Companhias telefônicas	Realiza a multiplexação de tráfego de múltiplos clientes sobre cabos de fibra ótica; fornece confiabilidade extra ao loop local e às redes da concessionária.

Tabela 11 – Uma visão geral dos serviços de rede digitais especializados.

SONET é um serviço óptico de alta velocidade de multiplexação que trabalha em cabeamento de fibra ótica. A SONET executa velocidades de até 13,22 gigabits. Ela é utilizada em redes de provedores de serviço de rede para transportar tráfego de múltiplos clientes que executam a diferentes velocidades. A SONET fornece uma maneira de as concessionárias aumentarem a confiabilidade de suas redes. O usuário final, no caso de ter aplicações críticas, pode contratar serviços de SONET entre suas localizações e a rede da companhia telefônica ou entre as próprias localizações da empresa (DODD, 2000).

Um resumo destes serviços está listado na Tabela 11.

Uma característica que todos os serviços citados têm em comum é que são transmissões digitais. Serviços analógicos não fornecem as velocidades mais altas, a exatidão e a confiabilidade disponível com comunicações digitais. As telecomunicações estão passando por um boom. Internet, acesso à Internet e telecomutação[6] são demandas que guiam os serviços. A própria Internet utiliza vastas quantidades da capacidade de rede.

A seguir vamos falar um pouco da Internet.

5.8.A Internet

[6] aplicações do tipo "trabalhe em casa".

A Internet é uma mídia que criou alterações fundamentais na natureza das comunicações, entretenimento e comércio. As empresas a utilizam para comunicar-se com fornecedores, colaboradores e clientes. Elas comercializam, fazem e recebem pedidos, fornecem serviço de atendimento ao cliente e orçamento de produtos e serviços pela Internet. As pessoas comumente utilizam a Internet para trocar correio eletrônico com pessoas, realizar operações bancárias, conduzir transações de estoque on-line, fazer reservas de viagem, compras e pesquisa.

Segundo Dodd (2000) “a Internet é uma conexão de múltiplas redes. As redes comunicam-se umas com as outras sobre um conjunto de protocolos padronizados, TCP/IP, que envia dados pela Internet divididos em *envelopes* chamados pacotes de dados”.

A World Wide Web é uma maneira de vincular documentos como home pages em computadores distantes entre si. Independentemente do tipo de computador utilizado, as pessoas com navegadores podem se conectar via web. É uma maneira “multimídia” de visualizar as informações pela Internet. Ela torna as informações disponíveis na forma de imagens gráficas, som, texto e vídeo.

Os indivíduos e organizações conectam suas localizações à Internet via muitos tipos de serviços de telecomunicações, como linhas analógicas, serviços DSL, ISDN, instalações de TV a cabo, entre outros já citados anteriormente. Os provedores de serviços da Internet (Internet Service Providers – IPSs) agregam tráfego de muitos usuários e lhes enviam linhas sobre de alta velocidade para o backbone da Internet. Os IPSs mantêm roteadores e servidores em seus locais. Os servidores realizam várias funções. Podem conter correio eletrônico de cliente, fornecer serviço de hospedagem para negócios e home pages para consumidores.

A tecnologia World Wide Web é utilizada por organizações comerciais para criar Extranets e Intranets. Na seção 5.8.1 faremos um breve resumo sobre o correio eletrônico. Na seção 5.8.2 falaremos um pouco sobre Intranets e Extranets.

5.8.1.Correio eletrônico

O correio eletrônico é o armazenamento e encaminhamento baseados em computador de mensagens baseadas em texto. O correio eletrônico é um fator importante no interesse dos consumidores de se conectar à Internet. Freqüentemente muitos consumidores adquirem computadores apenas para poder enviar mensagens de correio eletrônico para outras pessoas. As empresas utilizam o correio eletrônico para verificar status de projetos, acompanhar propostas e verificar status de pedidos. Além disso, trocam planilhas, apresentações em PowerPoint e documentos de processadores de texto através dos anexos de correio eletrônico.

O correio eletrônico mudou a maneira como consumidores e empresas se comunicam. De acordo com um estudo, depois que os usuários adquirem um correio eletrônico, é mais provável que eles enviem um correio eletrônico do que façam uma chamada telefônica interurbana. O correio eletrônico é mais rápido e menos evasivo que uma chamada telefônica. Uma chamada telefônica é uma interrupção, enquanto o correio eletrônico é armazenado e pode ser revisado quando conveniente (DODD, 2000).

Por fim, o correio eletrônico alterou significativamente a natureza da comunicação comercial. A seguinte citação de uma mensagem de correio eletrônico de Toni Proffetto, gerente administrativo da New Balance Athletic Shoe Company, exemplifica bem essa mudança: *conversarei com você amanhâ (via correio eletrônico evidentemente, já que ninguém mais conversa pessoalmente hoje em dia).*

5.8.2.Intranets e Extranets

As *intranets* são projetadas para serem redes abertas, seguras e internas, cujo software de navegação fornece acesso fácil de tipo apontar-e-clicar para usuários finais acessarem

informações em multimídia em sites da rede interna. Os sites de rede intranet podem ser estabelecidos em servidores de rede interna por uma empresa, suas uidades, departamentos e grupos de trabalho. Um dos atrativos das intranets é que sua tecnologia de tipo Internet as torna mais adaptáveis, bem como mais fáceis e mais baratas de serem desenvolvidas e utilizadas do que os sistemas tradicionais cliente/servidor ou sistemas legados de mainframe (O'BRIEN, 2002).

As *extranets* são redes que conectam alguns dos recursos intranet de uma empresa com outras organizações e indivíduos. As extranets permitem, por exemplo, que os clientes, fornecedores, consultores e outros acessem sites de rede intranet selecionados e bancos de dados de outras empresas. As organizações podem estabelecer extranets privadas entre elas, ou utilizar a Internet como parte das conexões de rede entre elas (O'BRIEN, 2002).

CONSIDERAÇÕES

Os principais componentes genéricos de uma rede de telecomunicações são (1) terminais, (2) processadores de telecomunicações, (3) canais de comunicações, (4) computadores, (5) software de telecomunicações. Existem muitos tipos diferentes de redes de telecomunicações como redes locais (LANs), redes metropolitanas (MANs) e redes geograficamente distribuídas (WANs). A maioria das WANs e LANs utiliza tecnologias cliente/servidor, computação em rede e Internet para formar intranets, extranets e outras redes interorganizacionais.

Os processadores de telecomunicações incluem modems, multiplexadores, processadores internetwork e vários dispositivos para ajudar a interconectar e aprimorar a capacidade e eficiência dos canais de telecomunicações. Os canais de telecomunicações

utilizam meios como fios de pares trançados, cabos coaxiais, cabos de fibra ótica, microonda terrestre, satélites de comunicações, sistemas de telefonia celular e tecnologias LAN sem fio. O software de telecomunicações, como os sistemas operacionais de rede, monitores de telecomunicações e navegadores de rede, controla e apóia a atividade de comunicações em uma rede de telecomunicações.

Neste capítulo abordamos também: largura de banda, topologias, arquitetura, protocolos e serviços especializados de redes de telecomunicações para ter um entendimento básico dessas tecnologias como forma de adquirir conhecimentos para embasar a escolha das tecnologias para troca de informações entre as UCTs e a sede do HEMOAM na capital Manaus.

A proposta de tecnologia para troca de dados entre as UCTs e a sede do HEMOAM é cada UCT se conectar a Internet através de um provedor (local ou da capital), utilizando o serviço de rede que disponha no município, e enviar os dados por e-mail. Na sede da Fundação HEMOAM, em Manaus, esses dados serão recebidos, importados para o sistema que existe (Sistema de Acompanhamento a Doação de Sangue - SAD) e enviada uma resposta por e-mail acusando o seu recebimento. Esta proposta será justificada no estudo de viabilidade apresentado no próximo capítulo.

No próximo capítulo abordaremos o estudo de caso, que é a modelagem do sistema para controlar a doação de sangue no interior do Estado do Amazonas.

CAPÍTULO 6 – ESTUDO DE CASO – MODELAGEM DO SISTEMA DE ACOMPANHAMENTO A DOAÇÃO DE SANGUE NO INTERIOR DO ESTADO DO AMAZONAS (SADI).

A Fundação de Hematologia e Hemoterapia do Amazonas - HEMOAM é a instituição no Amazonas que realiza os processos de coleta, tratamento e distribuição de sangue para transfusão sanguínea, na capital e no interior, atendendo em 100% a demanda do Estado. Integra a rede nacional de hemocentros e segue as diretrizes do Programa Nacional do Sangue e Hemoderivados, do Ministério da Saúde. Também é centro referencial na região Norte para diagnóstico e tratamento de doenças hematológicas, incluindo-se as hemoglobinopatias (anemia falciforme, talassemia), coagulopatias (hemofilia, doença de Von Willebrand) e doenças oncohematológicas (leucemias e linfomas, dentre outras).

O HEMOAM tem a missão de “garantir o suprimento de sangue e o atendimento hematológico à população do estado do Amazonas”, e para isso sua sede está localizada em Manaus e no interior do Estado atua com 47 Unidades de Coleta e Transfusão (UCTs). Essas Unidades de Coleta e Transfusão, foram implantadas para oferecer à população, do interior do Estado, sangue com qualidade assegurada.

Para garantir o cumprimento da missão institucional foi criado um Sistema informatizado que faz todo o acompanhamento das doações de sangue realizadas (SAD), na capital, desde o momento em que o doador chega à instituição até quem recebe o sangue

doado. Este sistema foi desenvolvido no paradigma estruturado, pela própria equipe de informática da instituição para controlar o grande volume de informações devido a quantidade de doações diárias recebidas e o volume de exames realizados por doação, além dos produtos gerados com o sangue e distribuídos por dia. Em média 200 doações/dia, 10 exames por doação e 200 produtos/dia distribuídos.

O problema é que o Sistema SAD é utilizado apenas na capital, na sede do HEMOAM; as UCTs dos municípios do interior ainda não estão informatizadas, sendo todo o trabalho feito manualmente e as informações anotadas em livros. Vale ressaltar ainda, que os exames para liberação da bolsa de sangue para transfusão são realizados na capital. Então, a UCT do interior tem que mandar as amostras de sangue e informações do doador para capital, aguardar os exames que serão realizados, e quando o resultado é finalizado essas informações são recebidas por telefone/fax para que a bolsa de sangue possa ser liberada para transfusão. Isto gera alguns problemas como: identificação do doador ilegível, pois as fichas que acompanham as amostras de sangue são escritas manualmente; dados incompletos, pois geralmente eles não mandam todas as informações necessárias; não podemos garantir segurança transfusional já que os resultados dos exames realizados são passados por telefone; o tempo de recebimento do laudo do resultado de exame pelo doador é longo chegando ser de até dois meses para algumas unidades.

A idéia para solução do problema é modelar um sistema de informação para as UCTs, que troque dados com o sistema que existe na sede do HEMOAM através da Internet. Esse sistema a ser modelado incluirá dados de cadastro do doador; cadastro de triagens; cadastro de doação com informações de numeração da bolsa de sangue e os exames realizados no sangue do doador com seus respectivos laudos. No futuro este sistema será implementado e implantado resolvendo, assim, o problema e agilizando todo esse processo.

Na próxima seção vamos conhecer um pouco mais sobre o HEMOAM fazendo um breve relato de sua história.

6.1.Um pouco da história da Fundação HEMOAM

A Fundação HEMOAM nasceu como Banco de Sangue do Hospital Universitário Getúlio Vargas, em 13 de agosto de 1982. Eram apenas 50 m^2 de área e pouco mais de 10 funcionários para atender exclusivamente os pacientes do hospital.

Com a criação do Programa Nacional do Sangue e Hemoderivados, o banco de sangue tornou-se hemocentro integrante da rede nacional. Em 1986 passou a suprir toda a cidade de Manaus com sangue de qualidade e tomou para si a tarefa de diagnosticar e acompanhar os portadores de hemofilia.

Conquistou, em 1987, sede própria e ampliou ainda mais suas atividades, passando a atender também os pacientes com doenças hematológicas graves. Ainda em 87 criou o Programa de Interiorização do Sangue e Hemoderivados, implantando Unidades de Coleta e Transfusão de Sangue com o objetivo de oferecer à população do interior do Amazonas sangue com qualidade assegurada.

Em 1989, a instituição foi transformada em fundação de direito privado e atualmente funciona em um espaço de 6 mil metros quadrados de área construída, estando em processo de reforma e ampliação que resultará em mais 4 mil metros quadrados que serão utilizados para melhor adequação de todos os serviços que desenvolve atualmente nas áreas de hemoterapia, hematologia, ensino e pesquisa, e análises clínicas.

Após implantação do seu Sistema da Qualidade e adequação às Normas ISO, o HEMOAM foi certificado pela ISO 9002 em dezembro de 2001, sendo o primeiro a obter a certificação entre os Estados do Norte, Nordeste e Centro-Oeste do país.

Na seção seguinte vamos conhecer o processo da doação sanguínea que servirá como base da informação para o sistema a ser modelado.

6.2.Conhecendo o processo da doação sanguínea

A doação sanguínea é o processo pelo qual uma pessoa se candidata para ser doadora de sangue. A partir daí, este candidato(a) passa por série de avaliações que determinam a qualidade de seu sangue para um receptor.

A doação pode ser voluntária, pré-depósito ou de reposição. A doação voluntária ocorre quando um doador vem doar espontaneamente e sem receptor específico; a doação de pré-depósito ocorre quando um doador vem doar para alguém que fará uma cirurgia; e a doação de reposição acontece quando já ocorreu a transfusão e o doador comparece para repor o estoque de sangue.

O candidato a doador se apresenta na recepção da HEMOAM para fazer o cadastro de seus dados pessoais. É feita, então, uma avaliação inicial para verificar se a pessoa está em condições de doar: verificando o peso, a altura e o hematócrito (diagnóstico de anemia). Se o hematócrito habilitar o candidato, ele preenche um questionário sobre: doenças sexualmente transmissíveis; doenças ocorridas nas últimas horas; se está ou não tomando algum medicamento; se possui algum vício; entre outras. É então encaminhado para uma entrevista com o médico, que analisará o questionário; validará seu aspecto físico; verificará sua pressão arterial – esta entrevista é chamada de *triagem*.

Se durante a *triagem*, o médico perceber que o candidato está faltando com a verdade sobre alguma pergunta ou apresentar algum problema de saúde, poderá impedi-lo de doar; garantindo assim a qualidade na seleção de candidatos. Nessa situação o candidato é classificado como *inapto*.

Se o candidato for aprovado na entrevista médica, é considerado apto a doar. É encaminhado para a sala de coleta do sangue. Uma amostra do sangue coletado é analisada pelos laboratórios de sorologia (análise para saber se há doenças no sangue) e imunohematologia (análise de ABO e fator Rh do sangue).

A bolsa de sangue coletada vai para o setor de *fracionamento* para gerar os produtos derivados do sangue. Quando os resultados de sorologia e imunologia estão concluídos e aprovados a bolsa de sangue é liberada para transfusão e o doador para novas doações (após 2 meses para homens, e 3 meses para mulheres).

No processo de doação de sangue existem alguns critérios que são avaliados para que o candidato possa doar seu sangue.

6.2.1. Critérios para doação de sangue

Em síntese todos os homens e mulheres podem ser doadores de sangue, desde que tenham entre 18 e 60 anos, pesem 50 quilos ou mais e estejam bem de saúde.

A qualidade de saúde de uma pessoa é representada pelos seguintes critérios:

i. teve hepatite depois dos 10 anos;

ii. tem comportamento sexual de risco;

iii. usa drogas;

iv. teve malária nos últimos 12 meses;

v. recebeu transfusão sanguínea;

vi. teve qualquer doença venérea nos últimos 12 meses;

vii. teve febre nos últimos 30 dias;

viii.está tomando algum medicamento.

Caso algum desses comportamentos for identificado, durante a entrevista médica, o candidato à doação não poderá doar e será classificado como *inapto*. Caso contrário será classificado como *apto* e poderá doar.

A doação de sangue poderá ser realizada na própria sede da Fundação HEMOAM em Manaus, mas existem outros locais de doação que veremos a seguir.

6.2.2.Locais de doação de sangue

A doação de sangue pode ser realizada na própria sede na capital Manaus. Existe também um ônibus de coleta, que visita os bairros da cidade. Caso o candidato a doação de sangue esteja no Estado do Amazonas, mas fora da cidade de Manaus, poderá fazer sua doação em qualquer Unidade de Coleta e Transfusão (UCT) implantada pelo HEMOAM no interior do Estado. Vamos conhecer, a seguir, os locais onde essas UCTs estão implantadas.

6.2.2.1.As Unidades de Coleta e Transfusão de Sangue

Dos 62 municípios do Amazonas, a Fundação HEMOAM possui 47 Unidades de Coleta e Transfusão implantadas são: Autazes, Amaturá, Anori, Atalaia do Norte, Alvarães, Apuí, Barcelos, Barreirinha, Benjamin Constant, Boa Vista do Ramos, Boca do Acre, Borba, Canutama, Carauari, Careiro Castanho, Coari, Codajás, Eirunepé, Envira, Fonte Boa, Humaitá, Ipixuna, Itapiranga, Itacoatiara, Juruá, Jutaí, Lábrea, Manacapuru, Manicoré, Maraã,

Maués, Nhamundá, Novo Airão, Nova Olinda do Norte, Novo Aripuanã, Parintins, Pauini, Presidente Figueiredo, São Gabriel da Cachoeira, São Paulo de Olivença, Santa Isabel do Rio Negro, Santo Antonio do Içá, Tabatinga, Tapauá, Tefé, Urucará, Urucurituba. Todas contam com estrutura hospitalar e demanda para serviços hemoterápicos.

A primeira Unidade de Coleta e Transfusão de Sangue implantada foi a de Coari.

Dentre as Unidades, acima citadas, as que possuem a maior demanda de coleta de sangue interna, média diária de entre 25 a 30 bolsas, são as Unidades de: Itacoatiara, Parintins, Tefé, Maués, Manacapuru, Humaitá, Apuí, Coari e Tabatinga.

Como já foi citado anteriormente, nenhuma das unidades de coleta e transfusão está informatizada. Apenas a capital (Manaus) possui um Sistema para Acompanhamento a Doação de sangue (SAD). Na próxima seção, descreveremos o SAD que será o sistema que trocará dados com o sistema a ser modelado neste capítulo (Sistema para Acompanhamento a Doação de sangue no Interior - SADI).

6.3.O Sistema de Acompanhamento a Doação de Sangue - SAD

O Sistema SAD foi desenvolvido pelos próprios técnicos da equipe de informática do HEMOAM, ele é uma atualização de um sistema, que existia anteriormente no HEMOAM, em *Clipper com DBF*. Foi desenvolvido no paradigma orientado a objetos, utilizando o ambiente visual (linguagem *Delphi*) e o sistema gerenciador de banco de dados (*Oracle*).

O SAD é o sistema responsável por todo o acompanhamento das informações sobre doação de sangue, dentre os vários setores internos (HEMOAM) envolvidos que são: Recepção e Triagem de doadores, Coleta da doação, Sorologia e Imunohematologia das amostras de sangue do doador, Fracionamento, Rotulagem e Distribuição da bolsa de sangue

para transfusão sanguínea. Sem a utilização desse sistema, não haveria como controlar o grande volume de informações diárias.

O sistema está dividido da seguinte maneira:

DOADOR

- Cadastrar e consultar de dados pessoais dos doadores de sangue.

TRIAGEM

- Cadastrar e consultar dados da triagem do doador, tais como sinais vitais (hematócrito, hemoglobina, pressão arterial, temperatura, peso, altura). Se o doador está apto à doação é cadastrada, caso contrário o motivo de sua inaptidão é cadastrado.
- Cadastrar e consultar dados do QBC[7] (caso exista para o doador).

IMUNOHEMATOLOGIA

- Receber tubos. É feito um cadastro quando os tubos chegam no laboratório para realização do exame.
- Checar tubos não recebidos. É verificado se algum tubo, das doações do dia, não deu entrada no laboratório.
- Lançar resultados dos exames. Os exames de imunohematologia são realizados e posteriormente seus resultados são cadastrados no sistema. São verificados o tipo ABO, fator Rh, e a presença de anticorpos.
- Definir conduta. Na imunohematologia existem dois tipos de conduta: bolsa liberada ou bolsa encaminhada para pesquisa. Quando há detecção, durante a realização do exame, de anticorpos que comprometam a transfusão, a bolsa não é liberada.

SOROLOGIA

- Receber tubos. É feito um cadastro quando os tubos chegam no laboratório para realização do exame.
- Checar tubos não recebidos. É verificado se algum tubo, das doações do dia, não deu entrada no laboratório.
- Lançar Sorologia básica. Lançamento de resultados de exames de patologias definidos pelo Ministério da Saúde. Por exemplo, hoje são realizados exames para detecção de HIV, Chagas, Hepatite B e C, Sífilis.
- Definir conduta da *sorologia básica*. Na sorologia básica existem três tipos de conduta: *liberada*, *inderterminada*, *descartada*. Quando a bolsa não apresenta nenhum resultado reativo para as patologias, a mesma é *liberada*. Em caso de não ser possível determinar um resultado reativo ou não, é lançada a conduta *indeterminada* para que posteriormente sejam realizadas repetições de exames até definição do resultado. Quando alguma patologia apresenta resultado reativo, a bolsa recebe a conduta de *descartada*, para evitar qualquer risco de liberação do produto possivelmente contaminado. Neste caso, é realizado o exame de *sorologia confirmatória*.
- Lançar *sorologia confirmatória*. Este exame comprova se o doador é portador ou não da patologia que apresentou resultado reativo na *sorologia básica*, liberando-o ou não, para futuras doações.

FRACIONAMENTO

- Gerar produtos. A bolsa de *sangue total* é fracionada em vários produtos distintos: concentrado de hemácias; concentrado de plaquetas; plasma. Registra-se no sistema os produtos gerados.

ROTULAGEM

---

[7] É um exame que indica se o doador está ou não com Malária, é realizado apenas se a pessoa esteve em alguma

- Rotular produtos. Quando os resultados do QBC, imunohematologia e sorologia são liberados, os produtos recebem uma etiqueta de identificação, gerada pelo sistema, e estão prontos para serem distribuídos. Quando um destes resultados impedir a liberação, os produtos são descartados.

DISTRIBUIÇÃO

- Distribuir produtos. A saída dos produtos é registrada no sistema, contendo informações de: data, hora, local de destino, e o nome do paciente específico quando houver doação de pré-depósito.

Atualmente esse sistema é utilizado apenas na sede do HEMOAM na capital. Nas UCTs do interior do Estado do Amazonas todo esse processo é feito manualmente. O cadastro do doador; a triagem; a coleta do sangue; os exames de imunohematologia; o fracionamento e a distribuição são realizados nas próprias unidades locais, com exceção dos exames de sorologia, e todas essas informações anotadas em livros.

Os exames sorológicos dos doadores do interior são feitos na capital. Semanalmente as UCTs enviam os tubos com as amostras para realização desses exames, juntamente com uma ficha contendo informações do doador e da doação. Quando os tubos chegam ao HEMOAM, são encaminhados para o laboratório de sorologia para realização dos exames. As fichas são digitadas no sistema.

As bolsas, coletadas no interior, são fracionadas e ficam aguardando a realização da sorologia para serem liberadas ou não.

Quando o resultado da sorologia está concluído, é registrado também no sistema. Um relatório contendo os resultados dos exames sorológicos é emitido e enviado via fax para cada UCT, para que a bolsa possa ser liberada para transfusão ou desprezada. Os exames de cada

---

zona endêmica.

doador são emitidos também pelo sistema, assinados e enviados via correio para cada UCT de destino.

O prazo para recebimento de resultados de exames sorológicos na capital é de uma semana. No interior, dependendo do local, o prazo pode durar até dois meses.

O sistema SAD utiliza a estrutura mostrada na Figura 21. O computador central fica na Gerência de Sistemas do HEMOAM. Ele é representado, para essa aplicação, por dois servidores principais: o servidor de Internet e servidor de Banco de Dados.

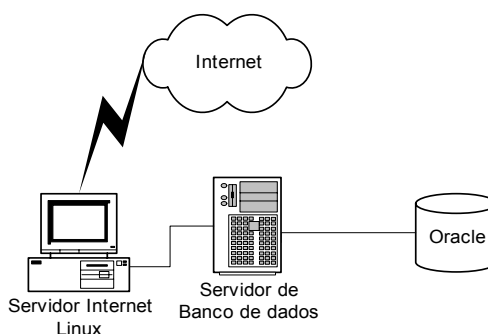



Figura 21 – Estrutura do computador central no HEMOAM.

O servidor de Internet é encarregado de prover acesso à Internet em computadores dentro do HEMOAM. A configuração de hardware do equipamento é Processador Intel Pentium 3; 750 MHz; 256 de memória RAM. O servidor de Internet utiliza o sistema operacional LINUX com dois serviços principais: Proxy e Firewall, permitindo acesso mais rápido aos dados e com segurança. O servidor de Internet está ligado ao Servidor de Banco de Dados, fazendo a triagem por acessos ao banco pelo próprio Firewall, diminuindo o risco de acesso indevidos por pessoas não autorizadas.

O servidor de Banco de Dados é responsável pelo gerenciamento do Banco de Dados Relacional do SAD, contém todos os dados dos doadores de sangue e de suas doações. Sua plataforma de hardware é: dois processadores Intel Pentium III Xeon® de 750 MHz; 2 GB de

memória RAM e 7 discos SCSI trabalhando em esquema de array de discos - RAID 5. A plataforma de software utilizada para gerenciar o Banco de Dados Relacional é o Oracle Server 8i. O servidor de Banco de Dados possui também outra finalidade que é prover o serviço de backup. Possui instalado um dispositivo de acesso a fitas padrão DAT de 20 a 40 GB.

A seguir, descreveremos o Sistema para Acompanhamento a Doação de sangue no Interior - SADI que será modelado logo nas próximas seções.

6.4.O Sistema de Acompanhamento a Doação de Sangue no Interior - SADI

O sistema proposto será uma aplicação local que trabalhará em conjunto com a aplicação cliente servidor SAD na capital, enviando dados de seu banco de dados local, no interior, para validação e inclusão no banco de dados principal da capital, e recebendo a base atualizada quando os exames forem concluídos. A Figura 22 ilustra cada um dos sistemas e seus locais de uso.

O arquivo que será enviado pela UCT conterá: os dados do cadastro do doador, sua triagem, os dados da coleta e os resultados dos exames da imunohematologia. Haverá no SADI uma opção de exportação desses dados no formato TXT. Em seguida, este arquivo será criptografado; compactado; e enviado por e-mail para o servidor de Internet localizado no HEMOAM. O servidor de Internet receberá este arquivo; descriptografará, e enviará o mesmo para o servidor de Banco de Dados, localizado também no HEMOAM, que gerará os registros na base principal do sistema SAD. Nas UCT's que não estiverem ligadas a um provedor de Internet a troca dados entre os dois sistemas será feita através de disquete.

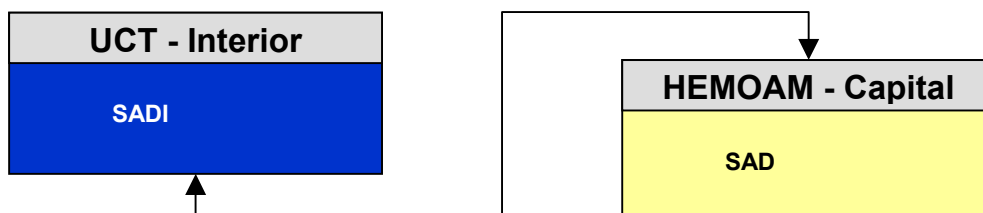



Figura 22 – Sistemas e seus locais de uso

O próximo passo é, realizar a sorologia no HEMOAM, e lançar seus resultados no SAD. O SAD também terá uma opção de exportação dos dados de resultados no formato TXT. Esse arquivo também será criptografado, compactado e enviado pela Internet, para a UCT no interior. Na UCT o arquivo será recebido, descriptografado, e o SADI gerará os registros na base de dados local. A justificativa para esta decisão e não o uso de uma outra solução será explicada no tópico do *estudo de viabilidade*.

Desta forma, o recebimento dos dados, tanto na capital como no interior, se tornará completo, mais rápido e seguro. A única coisa que não mudará é o tempo de chegada dos tubos no HEMOAM para realização dos exames sorológicos. Todo o restante do processo será agilizado quando o sistema SADI for implantado.

Cada UCT terá um computador PC, equipado com modem de transmissão de dados, ligado a uma linha telefônica, que fará a comunicação com o provedor de Internet, conforme mostra a Figura 23.

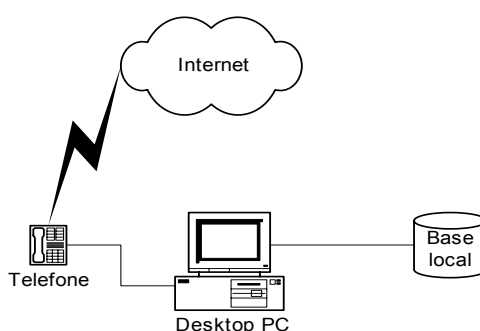



Figura 23 – Estrutura do computador local na UCT.

O acesso discado foi escolhido por atender a solução proposta, e também por permitir um custo de implantação e utilização relativamente pequeno. A comunicação não precisa ser diretamente on-line, o que é mais caro, mas é importante permitir o acesso relativamente rápido ao servidor do provedor de Internet.

6.4.1.A preocupação com a segurança dos dados do SADI

Por força da necessidade de sigilo definida em lei, existem alguns requisitos que devem ser satisfeitos a fim de garantir a segurança das informações do doador em todas as etapas de sua utilização, a saber:

i. controle de acesso: garantir que somente as entidades (pessoas ou outros componentes de software) autorizadas consigam acesso às informações do doador;
ii. autenticação: comprovar a identidade do usuário;
iii. confidencialidade: proteger informação armazenada ou transmitida contra divulgação às entidades não autorizadas;
iv. integridade: garantir que os dados não sejam modificados por entidades não autorizadas.

Durante o uso do módulo stand-alone, esses serviços de segurança não serão oferecidos pelo sistema e sim através da definição de procedimentos que restrinjam o acesso físico aos computadores das unidades de coleta e transfusão.

Durante o envio da mensagem será utilizado certificado digital para garantir a integridade, autenticidade e confidencialidade das informações.

O serviço de certificado digital recorre ao uso da criptografia mais especificamente a criptografia de chaves públicas.

A criptografia assegura a proteção da informação usando uma função matemática ou algoritmo para cifrar e decifrar mensagens. Na verdade existem dois tipos de algoritmos criptográficos

i. os simétricos, ou convencionais, em que só existe uma chave para criptografar (cifrar) e descriptografar (decifrar);

ii. os assimétricos, ou de chave pública, em que existem duas chaves diferentes, matematicamente relacionadas (Pública e Privada).

O conceito básico da relação criptográfica entre uma Chave Privada e uma Chave Pública é simples: O que uma chave cifra "só" a outra decifra. Ou seja, qualquer informação cifrada pela chave privada só é decifrada pela chave pública correspondente ou qualquer informação cifrada pela chave pública só é decifrada pela chave privada correspondente. Vamos descrever, a seguir, como os conceitos de criptografia e certificado digital se aplicam ao SADI.

6.4.1.1.Confidencialidade

Num sistema de criptografia por chave pública, a informação é cifrada com a chave pública da pessoa a quem é dirigida a mensagem e só a pessoa que tem a chave privada correspondente (o destinatário correto) é que consegue decifrar a mensagem.

Com o sistema de chave pública, tanto o emissor como o destinatário podem, de uma forma segura, trocar informação privada numa transação eletrônica. No nosso cenário, para enviar as informações do doador de forma segura o usuário da unidade de coleta e transfusão utiliza a chave pública do destinatário para cifrar os dados. Quando o destinatário recebe os dados, pode decifrar essa informação com a sua chave privada, e assim obter acesso aos dados do doador em formato legível. Enquanto o destinatário mantiver a sua chave privada segura a informação transacionada muito dificilmente será acessada por entidade não autorizada.

6.4.1.2.Integridade e autenticidade

Os serviços de integridade e autenticidade são implementados através da assinatura digital. De maneira análoga à assinatura de documentos em cartórios, o serviço de assinatura digital garante que a informação em questão é autentica e não sofreu nenhuma adulteração.

Para criar uma assinatura digital, figura 24, é necessário primeiro que o emissor da mensagem gere uma versão reduzida da mensagem (160 bits, por exemplo), conhecida por código *Hash* ou código da mensagem. Este código, gerado por algoritmos públicos (SHA-1, MD5, etc), é único para a mensagem original. Basta alterar ligeiramente o texto original para que o código então gerado seja completamente diferente.

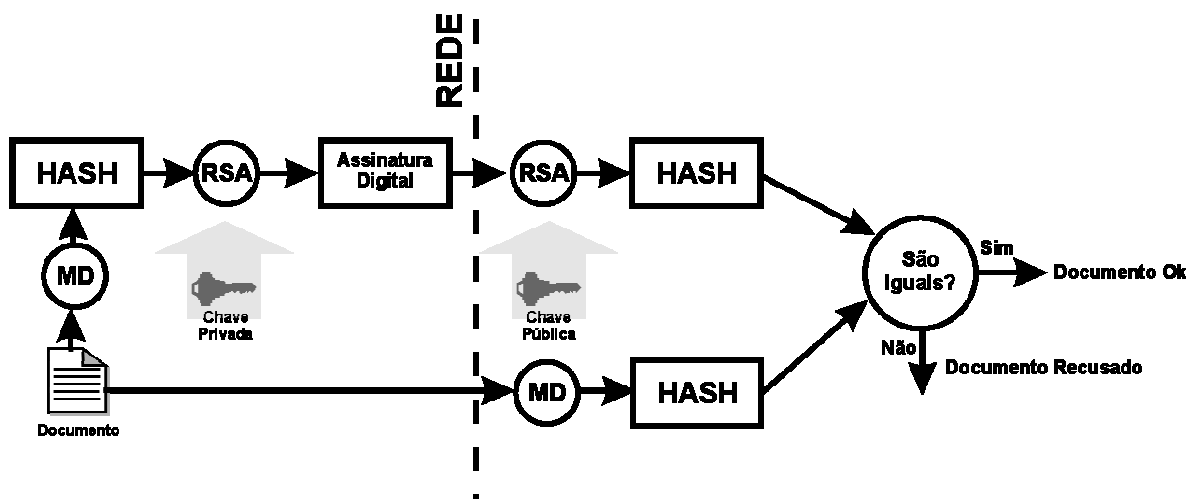



Figura 24 – Criação e verificação da assinatura digital.

Após a recepção da mensagem, o receptor decifra o código *Hash* recebido com a chave pública do emissor. Para verificar a exatidão da assinatura digital e a integridade da informação, o receptor gera um código *Hash* com a mensagem original que recebeu, e compara este código com o que obteve da decifragem da assinatura digital recebida. Se o receptor validar positivamente a assinatura digital com a chave pública do emissor, temos a garantia de que a mensagem foi enviada pelo dono da chave privada e que não foi alterada durante a transmissão.

6.4.1.3.Certificado digital

Existe a necessidade de garantir que o par de chaves (pública e privada) pertence realmente a um determinado emissor. Isso é feito através da utilização de Certificados Digitais, emitidos por uma terceira parte confiável.

Antes de duas partes trocarem informação usando criptografia de chave pública, cada uma delas, deseja autenticar a identidade da outra parte. Uma forma de assegurar a autenticação de cada uma das partes pode ser alcançada através dos serviços de uma terceira parte que emite e regule os serviços de gestão de certificados digitais.

Um certificado digital é um documento eletrônico assinado digitalmente, emitido por uma terceira parte de confiança, denominada Autoridade Certificadora.

Usando metodologias, processos e critérios bem definidos e públicos, a Autoridade Certificadora regula a gestão dos certificados através da emissão, renovação e revogação dos mesmos, por aprovação individual.

Para criar um certificado digital, a Autoridade Certificadora cria um código *Hash* com, entre outros dados, a informação da identidade do utilizador e a sua chave pública. A Autoridade Certificadora assina esta informação ao utilizar a sua chave privada, criando um código *Hash* cifrado. Esta informação é incluída no certificado a emitir. Se a informação da identidade do usuário, ou a chave pública contida no certificado digital for alterada de qualquer forma, o certificado é detectado como inválido.

Para confirmar a integridade de um certificado digital, o receptor:

i. recria o código *Hash* usando o mesmo algoritmo e informação que a Autoridade Certificadora utilizou na criação do *Hash* original para o certificado em questão;

ii. decifra o *Hash* existente no Certificado Digital com a Chave Pública da Autoridade Certificadora;

iii. compara os dois valores obtidos e se estes forem iguais, o Certificado Digital apresenta-se como íntegro. Caso contrário, o Certificado Digital sofreu alterações e é inválido.

6.4.2.O Processo de Desenvolvimento do SADI

O processo de desenvolvimento a ser utilizado é baseado no Rational Unified Process. Possui definição de atividades, responsabilidades, artefatos de entrada e saída, e ferramentas que serão utilizadas; tem ênfase na criação e manutenção de modelos da UML.

O armazenamento dos documentos gerados ao longo do processo é uma preocupação deste processo de desenvolvimento. Os documentos em papel são organizados em pastas e guardados no armário da Gerência de Sistemas. Os arquivos de computador (doc, txt, etc.) são armazenados no servidor de arquivos também situado na Gerência de Sistemas. Quanto à segurança desses arquivos, possuímos um sistema de backup em fitas DAT (semanal e mensal).

O processo de desenvolvimento utiliza as etapas de: análise, projeto, implementação, verificação e validação, implantação e manutenção. Cada etapa é estruturada com um conjunto de atividades. A Figura 25 ilustra as etapas e a Figura 26 os modelos que serão utilizados.

A UML não determina uma ordem predefinida dos diagramas que devem ser modelados primeiramente. Esta ordem é determinada pela preferência do desenvolvedor e/ou processo que esteja sendo usado. Alguns desenvolvedores iniciam a modelagem do sistema pela criação das classes, outros pelos casos de uso. No nosso processo de desenvolvimento iniciaremos a modelagem a partir dos casos de uso.

A seguir descreveremos cada etapa sucintamente. **Ressaltamos que o escopo deste trabalho é apenas "modelagem do sistema SADI", portanto o estudo de caso será desenvolvido até a etapa de projeto.** As demais etapas apenas serão descritas.

A *etapa de análise* é de responsabilidade do analista de sistemas e tem como atividades principais:

i. levantamento das necessidades;
ii. estudo de viabilidade geral do sistema;
iii. modelagem de negócios;
iv. modelagem do domínio do problema.

Levantamento das necessidades – investigação do domínio do problema. Neste levantamento esta incluída a validação de requisitos. São utilizadas técnicas para colher informações dos vários tipos de usuários do sistema. Inicialmente são realizadas entrevistas nas quais um documento é redigido pelo analista de sistemas sobre as necessidades do usuário e ambas as partes assinam o documento de comum acordo. Estas entrevistas são divididas em dois tipos internas (usuários) externas (gerentes de setor). Após o término das entrevistas individuais é realizada uma reunião que auxiliará na validação dos requisitos extraídos junto aos usuários. No final destas reuniões onde participarão usuários e gerentes é elaborada uma ata onde todos assinam. Em seguida são realizados laboratórios em cada setor participante do processo onde são estudas as atividades e os documentos relacionados. A partir daí é elaborado o escopo do sistema.

Estudo de viabilidade geral do sistema – investigação que determinará os requisitos de recursos, custos, benefícios e viabilidade do projeto proposto.

Modelagem de Negócios – na modelagem de negócio será feita a análise de requisitos, a especificação de requisitos usando-se casos de uso, atores e cenários.

Modelagem do Domínio do Problema – identificação de classes do domínio do problema e seus atributos.

Cada uma destas atividades principais são desmembradas em outras atividades, mostradas na tabela 12, a seguir, onde há uma pessoa responsável em realizar cada atividade, que ferramenta é utilizada no desenvolvimento da atividade, e os artefatos de entrada (necessário para realizar a atividade) e saída (produto da realização da atividade).

Na etapa de Análise, conforme mostra a tabela 12, todas as atividades são de responsabilidade do analista de sistemas. A primeira atividade é *a identificação do negócio do sistema*. Processos e requisitos de negócio são descobertos e expressos em casos de uso. Compreender os requisitos, inclui, em parte, compreender os processos do domínio e o ambiente externo – fatores externos que participam dos processos. Esses processos do domínio são expressos em casos de uso. Passo preliminar útil na descrição dos requisitos do sistema.

A atividade *identificação do negócio do sistema* tem como artefato de entrada o *escopo do sistema*, que é descoberto a partir do levantamento de requisitos, e produz como artefato de saída a *lista de casos de uso*. O método para identificar casos de uso é baseado em atores, no qual primeiramente são identificados os atores relacionados com o sistema ou a organização e em seguida, para cada ator, são identificados os processos que eles iniciam ou dos quais eles participam. Gerando então a lista de casos de uso. A partir de um caso de uso é possível chegar a outro.

Etapas	Atividades	Responsável	Ferramenta	Artefato entrada	Artefato saída
Análise	Identificação do negócio sistema	Analista de Sistemas	Processador texto	Escopo do sistema	Lista de casos de uso
Análise	Modelagem do negócio sistema	Analista de Sistemas	Rational Rose	Lista de casos de uso	Diagrama de casos de uso
Análise	Criação dos cenários do negócio do sistema	Analista de Sistemas	Rational Rose	Diagrama de casos de uso	Cenários principal e alternativo
Análise	Validação do negócio sistema com usuários	Analista de Sistemas	Reunião	Diagrama de casos de uso e cenários principal e alternativo	Ata da reunião com data e assinatura
Análise	Identificação das classes do domínio do problema e seus atributos	Analista de Sistemas	Processador texto	Diagrama de casos de uso	Lista de classes com seus atributos
Análise	Modelagem do domínio do problema	Analista de Sistemas	Rational Rose	Lista de classes com seus atributos	Diagrama de classes com atributos e relacionamentos

Tabela 12 – Uma visão geral das atividades, responsabilidades, ferramentas utilizadas e artefatos da etapa de Análise do Processo de Desenvolvimento do sistema SADI.

A partir da *lista de casos de uso* (artefato de entrada) é realizada a atividade de *modelagem do negócio do sistema*, que utiliza como ferramenta, para produzir os *diagramas de casos de uso* (artefato de saída), o software Rational Rose.

Uma vez identificados os casos de uso, nos concentraremos em descrever os cenários principais. A partir dos cenários principais, identificamos e descrevemos os cenários alternativos.

Vale ressaltar aqui, que apenas nos preocupamos em escrever os casos de uso desta forma (formato alto nível). Não nos preocupamos em categorizá-los (primários, secundários ou opcionais) como explicam alguns livros de padrões sobre a UML.

É comum durante a descrição percebermos a necessidade de cenários comuns a outros casos de uso. Então são descobertos os casos de uso de inclusão. Da mesma forma, casos de uso muito extensos, seja quanto ao cenário principal ou quanto aos cenários alternativos, devem ser divididos em casos de extensão.

A preocupação com a escrita de casos de uso mais críticos, influentes e de maior risco, seja no formato *essencial expandido*, ou em outro formato, conforme descrito no livro de Craig Larman (LARMAN, 1999) não faz parte do escopo deste trabalho.

Os *casos de uso* possuem um enfoque conceitual dentro da modelagem em UML. A implementação de um caso de uso ocorre através dos *diagramas de interação* (etapa de projeto). Uma descrição mais detalhada sobre como descrevemos os casos de uso, é abordada no tópico do estudo de caso "modelagem de negócio do sistema SADI" mais a diante, no próximo capítulo.

A partir dos *diagramas de casos de uso* (artefato de entrada) é realizada a atividade de *criação dos cenários do negócio do sistema*, que utiliza o software Rational Rose para produzir os *cenários principal e alternativo* (artefato de saída).

A partir dos *diagramas de caso de uso e dos cenários principal e alternativo* (artefato de entrada) é realizada a atividade de *validação do negócio do sistema com nossos usuários*, através de reunião. Esta reunião gerará uma ata com o resumo de tudo que foi discutido (a ata é o artefato de saída desta atividade).

A partir dos *diagramas de caso de uso* (artefato de entrada) é realizada a atividade de *identificação das classes do domínio do problema com seus atributos*. O analista utiliza um processador de texto e lista as classes encontradas, e seus atributos, a partir dos diagramas de caso de uso. O artefato de saída é esta *lista*.

A partir da lista criada (artefato de entrada), o analista de sistemas realiza a atividade de *modelagem do domínio do problema*. Usando a ferramenta Rational Rose este cria o

diagrama de classes com os atributos e relacionamentos (artefato de saída). Neste momento é finalizada a etapa de análise e a partir daí iniciada a etapa de projeto.

Na *etapa de projeto* os diagramas da análise continuam válidos, porém são revisados. Novos diagramas serão modelados refletindo requisitos de implementação. Tem as seguintes atividades principais:

i. decisões de projeto;
ii. modelagem comportamental;
iii. modelagem arquitetural.

Decisões de projeto – são definidos o tipo de banco de dados e linguagem de programação que serão utilizados, os mecanismos de acesso a atributos, a plataforma de implantação, número máximo de usuários que poderão acessar o sistema simultaneamente, as interfaces com o usuários, etc.

Modelagem Comportamental – descrição do comportamento do sistema, mostrando como são realizadas as interações entre os objetos, quais os estados nos quais um objeto pode estar e quais operações ele pode realizar. Para operações complexas, são mostradas as ações e atividades que as compõem. Os diagramas elaborados são:

i. diagrama de interação;
ii. diagrama de classe;
iii. diagrama de estado;
iv. diagrama de atividades.

Modelagem Arquitetural - o objetivo da modelagem arquitetural é descrever a arquitetura física do sistema. Seu foco é o mapeamento da estrutura lógica de classes para uma arquitetura física em termos de componentes e nós de processamento. A arquitetura física descreve a decomposição detalhada do hardware e software que compõem a implementação do sistema. Os diagramas utilizados são os diagramas de implementação:

i. diagrama de componentes;

ii. diagrama de implantação.

Assim como na etapa de análise, na etapa de projeto cada uma destas atividades principais são desmembradas em outras atividades, mostradas na tabela 13.

Conforme mostrado na tabela 13, as atividades são de responsabilidade do analista de sistemas, mas existe a participação do Administrador de Banco de Dados (DBA), e do Gerente de Sistemas. A primeira atividade é *a tomada de decisões de projeto*, onde são elaborados documentos com definição do tipo de banco de dados a ser utilizado no projeto do sistema, a linguagem de programação que será utilizada para implementar as decisões modeladas, entre outros requisitos. Esta atividade produz estes *documentos elaborados* como artefatos de saída.

A partir das decisões anteriores, parte-se para realização da segunda atividade de projeto que é a *construção da estrutura do menu*, a partir da *lista de casos de uso* elaborada na etapa de análise. Com o uso de um processador de texto tem-se a *estrutura de menu* como artefato de saída desta atividade.

A próxima atividade da etapa de projeto é o *estudo do fluxo da informação*, realizado com base no *escopo do sistema* levantado na etapa de análise (artefato de entrada). O *diagrama de atividades* (artefato de saída) é elaborado nesta atividade.

Etapas	**Atividades**	**Responsável**	**Ferramenta**	**Artefato entrada**	**Artefato saída**
Projeto	Decisões de Projeto	Analista de sistemas; DBA; gerência de sistemas	Processador de texto	-	Documento onde estarão definidos: tipo de BD; LP; plataforma implantação etc.

Projeto	Construção da estrutura do menu	Analista de Sistemas	Processador texto	Lista de casos de uso	Estrutura do menu
Projeto	Estudo do fluxo da informação	Analista de Sistemas	Rational Rose	Escopo sistema	Diagrama de atividades
Projeto	Validação fluxo sistema com usuários	Analista de Sistemas	Reunião	Diagrama de atividades	Ata da reunião com data e assinatura
Projeto	Modelar interação entre objetos	Analista de Sistemas	Rational Rose	Casos de uso	Diagrama de sequência
Projeto	Completar o diagrama de classes com as operações descobertas	Analista de sistemas	Rational Rose	Diagramas de sequência	Diagrama de Classes com as operações
Projeto	Modelar o comportamento dos objetos com comportamento dinâmico	Analista de Sistemas	Rational Rose	Escopo Sistema e Diagrama de Classes com as operações	Diagramas de estados
Projeto	Modelar arquitetura física em termos de componentes	Analista de Sistemas	Rational Rose	Diagrama de Classes com as operações	Diagrama de Componentes
Projeto	Modelar arquitetura física em termos de nós processamento	Analista de Sistemas	Rational Rose	Diagrama de Classes com as operações	Diagrama de implantação

Tabela 13 – Uma visão geral das atividades, responsabilidades, ferramentas utilizadas e artefatos da etapa de Projeto do Processo de Desenvolvimento do sistema SADI.

Depois de elaborado o diagrama de atividades, o analista de sistemas deverá *validar o fluxo do sistema com os usuários*. Esta validação será realizada por meio de reunião(ões) onde o diagrama elaborado será discutido. Após a(s) reunião(ões) uma *ata* será elaborada com o que ficou definido. Os participantes da(s) reunião(ões) assinarão esta ata.

A seguir será *modelada a interação entre os objetos*. Cada caso de uso encontrado (artefato de entrada) gerará um *diagrama de seqüência* (artefato de saída) para cada cenário principal. Os cenários alternativos serão explicados nestes diagramas por meio de notas.

Durante a criação dos diagramas de seqüência as operações das classes são descobertas. Então a próxima atividade é *completar o diagrama de classes com essas operações descobertas*.

De posse do *diagrama de classes com as operações*, o analista de sistemas parte para modelar o comportamento de objetos que possuem comportamento dinâmico. Além do diagrama de classes com as operações, o analista também utiliza o *escopo do sistema* como artefatos de entrada para elaborar os *diagramas de estados* (artefatos de saída). Ele também modela a arquitetura física em termos de componentes e nós de processamento, criando os *diagramas de componentes e de implantação*.

Sempre que novas classes e operações forem descobertas, os diagramas de casos de uso serão revisados.

Com estas atividades a etapa de projeto está concluída.

As próximas etapas: implementação, verificação e validação, implantação e manutenção não fazem parte do escopo deste trabalho. Desta forma, elas serão descritas de forma resumida.

A *etapa de implementação* é de responsabilidade do programador, deve seguir os modelos criados nas etapas de análise e projeto e tem como atividade apenas a codificação do sistema na linguagem de programação escolhida.

A *etapa de verificação e validação* é de responsabilidade dos analistas e programadores, deve seguir os modelos criados pelas fases anteriores, deve fazer uso de técnicas de verificação e validação e tem como atividades:

i. construção de um plano de verificação e validação;

ii. testes de corretude do sistema através do projeto;

iii. testes de funcionalidade e aceitação dos usuários.

A *etapa de implantação* é de responsabilidade dos analistas, programadores e pessoal de suporte, deve seguir os modelos criados pelas fases anteriores e tem como atividades:

i. plano de implantação de sistemas;
ii. preparação do ambiente de produção;
iii. treinamento de usuários.

A *etapa de manutenção* é de responsabilidade dos analistas, programadores e pessoal de suporte, deve seguir os modelos criados pelas fases anteriores, deve assegurar o correto funcionamento do sistema e tem como atividades:

i. controle de atualização de versões;
ii. tipos de manutenção que serão realizadas: corretiva, adaptativa.

Para cada atividade a ser realizada, em cada umas das etapas citadas anteriormente haverá recursos, artefatos e restrições. O artefato é o que é consumido e produzido pelas atividades, são os produtos de entrada e os produtos de saída obtidos ao término da atividade. Os recursos são alocados para a realização da atividade como: cargo (responsável pela atividade), ferramentas (utilizadas durante a atividade). As restrições são condições que as atividades devem satisfazer antes e depois de ser executada.

A seguir iniciaremos a etapa de análise com a descrição do estudo de caso do sistema SADI e em seguida a etapa de projeto.

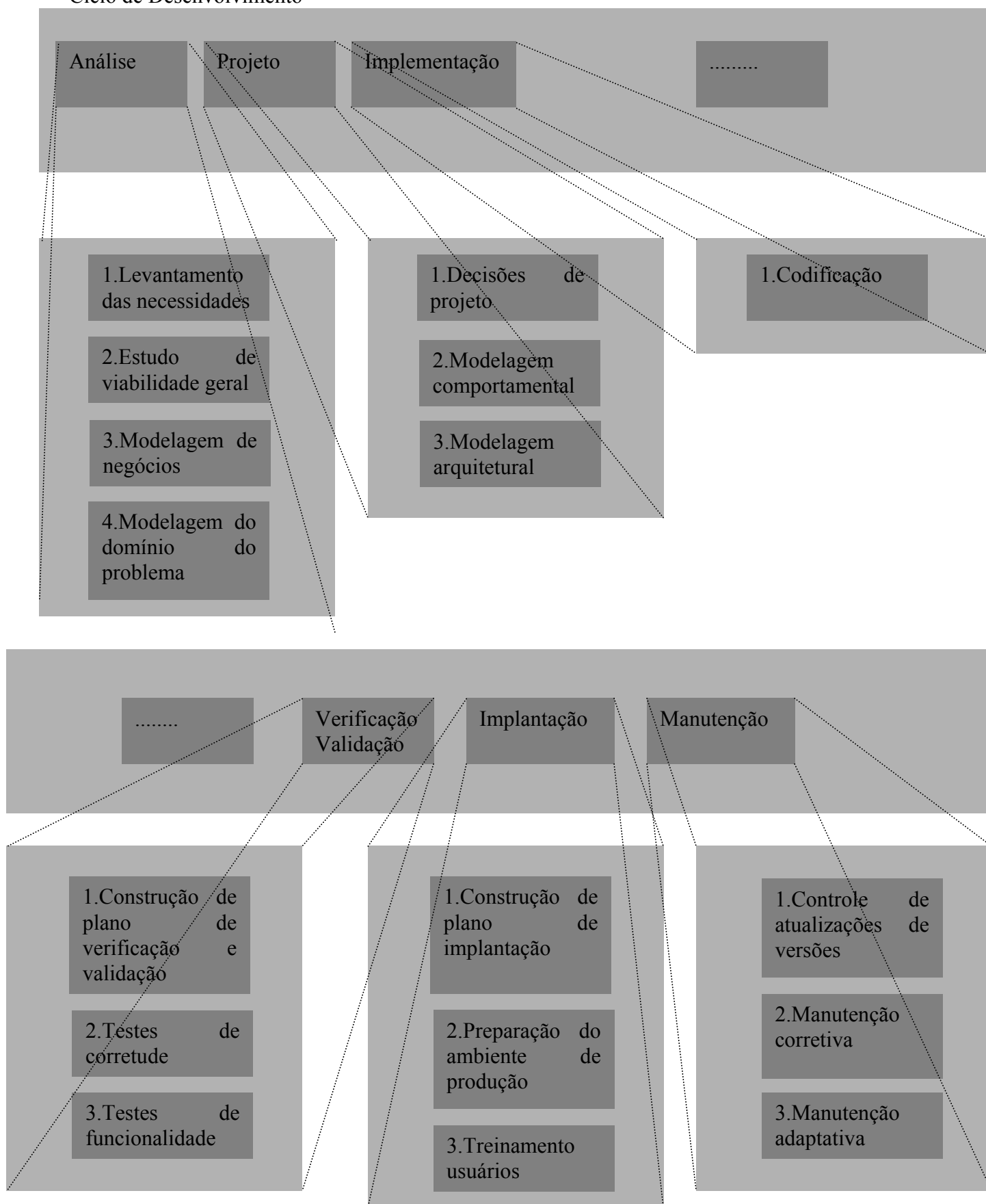



Figura 25 – Atividades do Processo de Desenvolvimento do Hemoam.

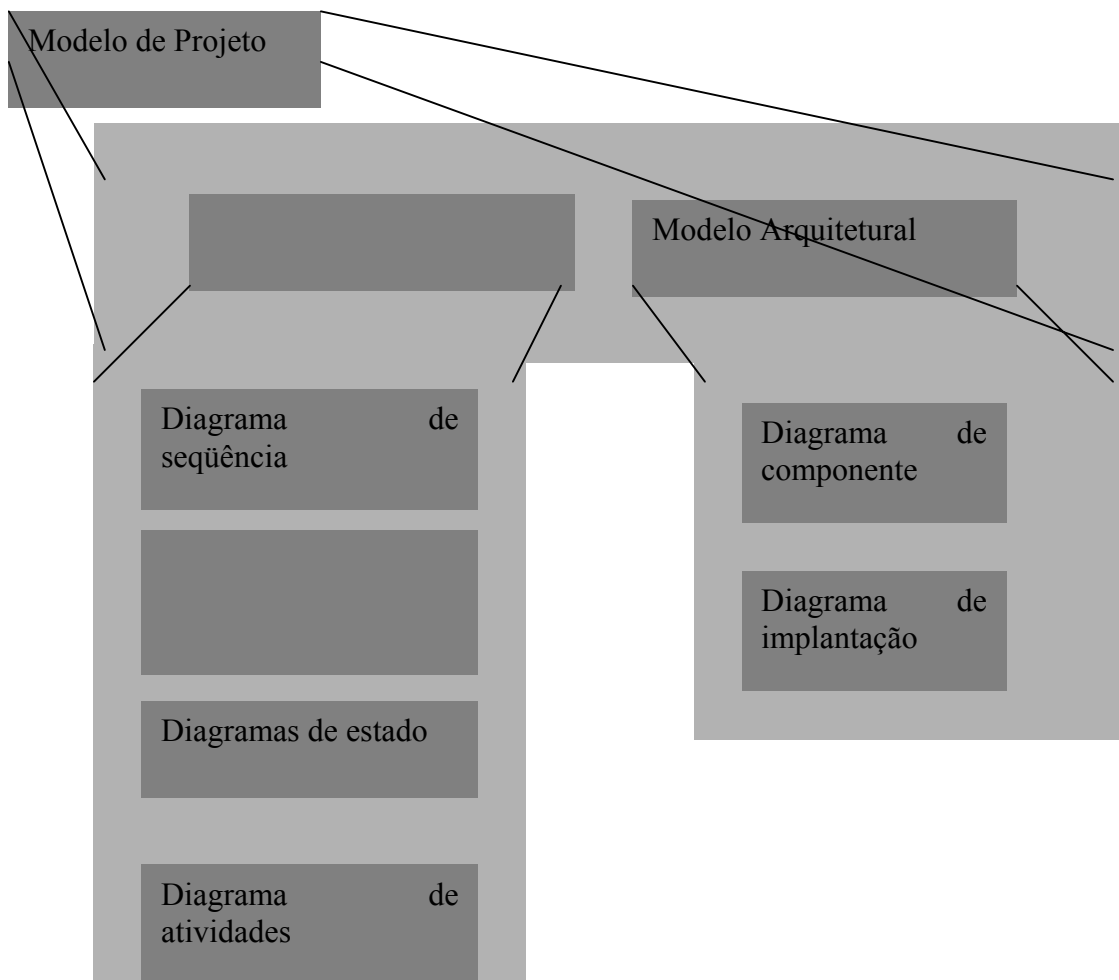
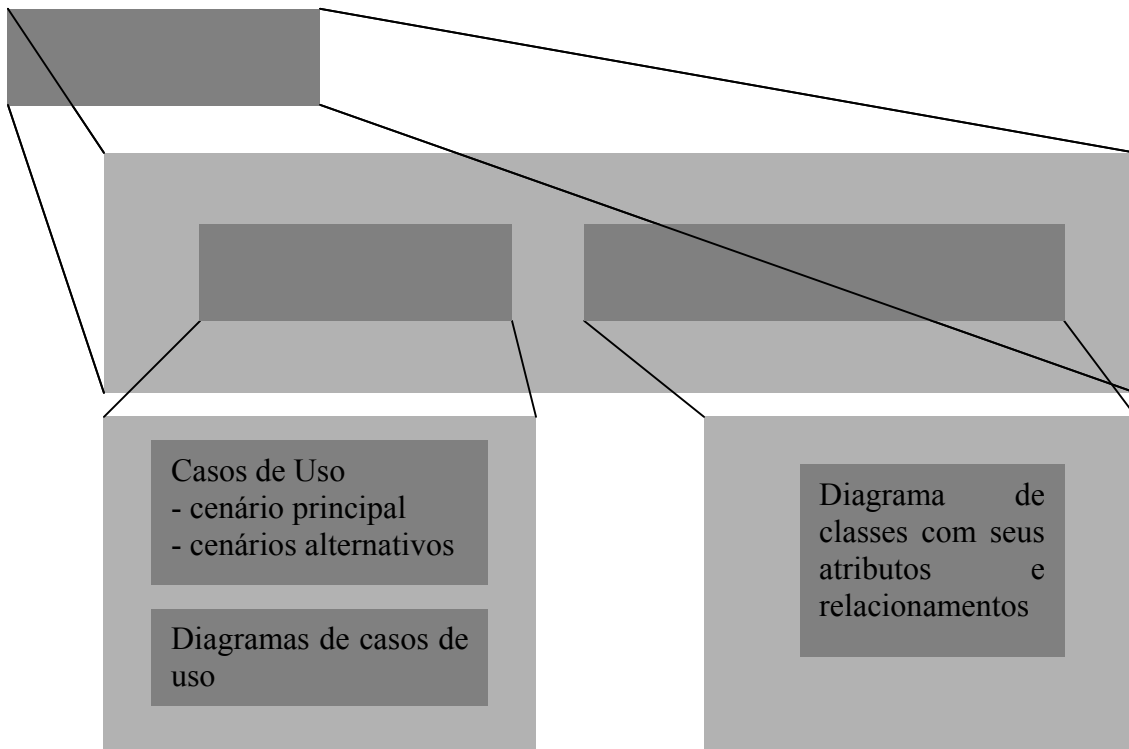



Figura 26 – Modelos utilizados no Processo de Desenvolvimento do Hemoam .

6.4.2.1.Descrição do Estudo de caso - Sistema SADI

O candidato à doação apresenta-se na Recepção de doadores da Unidade de Coleta e Transfusão (UCT).

Identifica-se. Sendo um novo candidato, seus dados pessoais deverão ser cadastrados e impressa uma ficha para doação. Se o candidato já tinha cadastro, seus dados cadastrais deverão ser conferidos, atualizados se houver necessidade (troca de endereço, por exemplo) e suas informações anteriores checadas, objetivando definir se o candidato continua apto ou não.

Sendo o doador apto, o mesmo é inserido no circuito da doação, caso contrário, deve ser registrado o motivo de inaptidão, e o candidato é dispensado.

A ficha do doador contém informações de seu cadastro e da doação e acompanhará o candidato em todo o circuito da doação.

O candidato apto na Recepção é submetido ao Hematócrito, onde é medido e pesado. Se for detectado que o mesmo está abaixo do peso mínimo exigido, não lhe é permitido continuar no Circuito. O peso mínimo é de 50kg, no entanto, em algumas situações, o médico faz um cálculo – verificando a proporção entre peso e altura, que pode liberar ou não para o passo seguinte o candidato que está fisicamente abaixo do peso mínimo exigido. Depende da avaliação médica. Aprovado no exame de altura/peso, é colhido um pouco de sangue do seu dedo, para a realização do teste chamado *hematócrito*, cujo objetivo é determinar se o candidato está ou não com anemia. A duração do teste é cerca de 20min.

Enquanto o hematócrito é realizado, o candidato preenche um questionário com perguntas sobre seus hábitos pessoais, sexuais, doenças já contraídas, etc.

Concluindo-se o exame de hematócrito, o resultado da leitura é anotado na ficha de doação, bem como o peso e a altura do doador.

Não constatada anemia ou deficiência de peso, o candidato é encaminhado para a Triagem (entrevista), levando consigo a ficha de doação e o questionário preenchido e assinado.

A entrevista é realizada por um médico ou um técnico de hemoterapia. O questionário preenchido, as informações de peso, altura e hematócrito e os sinais vitais do doador são avaliados. Se, durante a entrevista, o técnico perceber que o candidato está mentindo sobre alguma pergunta, poderá impossibilitá-lo de doar sangue.

Se o candidato for considerado inapto, será informado de que não poderá doar sangue e será despachado. Se considerado apto, o técnico seleciona uma bolsa de sangue – os kits são previamente preparados e cada um contém: uma bolsa e 3 tubos, etiquetados com o mesmo código de barras, contendo um número de doação e os tubos de coleta de sangue para exame no laboratório. O técnico registra no sistema todas informações dessa triagem, entrega ao doador seu "kit doação" e o encaminha ao lanche para tomar um suco e, a seguir, à sala de coleta.

Na sala de coleta, o sangue é colhido na bolsa e tubos entregues ao doador. A bolsa é encaminhada ao setor de Fracionamento, para que o sangue seja imediatamente fracionado, senão as células morrem. Um dos tubos é encaminhado ao laboratório de Imunologia e os outros dois tubos para a capital (HEMOAM) para realização do exame de Sorologia. Ambos laboratórios farão a análise da amostra ali contida.

Chegando ao fracionamento, independente da conclusão dos exames a serem realizados, a bolsa de sangue total é pesada e fracionada e os componentes do sangue são obtidos, separadamente. O fracionamento consiste em centrifugar a bolsa de sangue, repetidas vezes. De cada vez, o produto almejado é separado do sangue total. Assim, são produzidos plasma, concentrado de hemácias, plaquetas, crio precipitado do fator VIII, etc.

No processo de fracionamento, alguns produtos podem perecer, seja porque ocorreu uma hemólise (estourou na centrífuga) ou por estar o sangue lipêmico, etc.

Os produtos que lograram êxito em sua geração deverão ser registrados no sistema, ficando com o status de produtos gerados. São armazenados em condições apropriadas, aguardando a conclusão dos exames. Para os que não chegaram ao final do fracionamento, também deverá ser registrada a *intercorrência* havida.

Os exames de imunologia são realizados geralmente no mesmo dia, mas podem ser feitos no dia seguinte ao da coleta. Chegando tubo no laboratório de Imunologia, uma lista de trabalho é preparada com a discriminação das amostras que deverão ser examinadas. São realizados exames para identificar a tipagem sanguínea do doador, RH, a presença de anticorpos irregulares. A imunologia registra no sistema resultados dos exames e uma conduta final sobre os produtos, indicando se poderão ser transfundidos ou não, considerando os exames ali realizados. Após o lançamento da conduta no sistema, o funcionário o laboratório de imuno gera os arquivos de exportação (TXT), com os dados de cadastro do doador, triagem, coleta e imunologia, naquele dia ou período em que houve as doações. Esses arquivos são enviados por e-mail para capital.

As amostras que são enviadas para capital, para realização dos exames sorológicos, também podem chegar no mesmo dia, vai depender da distância do município. O tempo máximo de chegada da amostra, do município mais distante, é uma semana.

Na capital, no laboratório de sorologia, o técnico importa os arquivos da UCT (sistema capital). Chegando o tubo no laboratório uma lista de trabalho é preparada com a discriminação das amostras que deverão ser examinadas.

Atualmente, em face da legislação vigente, são realizados oito exames diferentes nas amostras, exames esses que objetivam diagnosticar a presença de doenças como sífilis, hepatite, aids, doença de chagas e outras mazelas que podem acometer o doador.

Obtidos os resultados, serão registrados no sistema e determinado se, em relação à sorologia, os produtos daquela bolsa podem ser liberados ou não, conduta sorologia. Esta conduta também é registrada no sistema. O técnico gera o arquivo de exportação com esses resultados e envia por e-mail para a UCT do interior.

Quando os resultados de exames da sorologia chegam na UCT, o técnico do laboratório importa esses resultados para a base local e avisa ao pessoal da rotulagem que os dados chegaram.

O técnico da rotulagem retira os produtos da câmara fria, verifica a validade do produto, verifica a conduta da bolsa para ver se está liberada. Estando tudo em ordem, os produtos gerados a partir daquela bolsa são considerados liberados para distribuição, são rotulados. Se a conduta da bolsa estiver desprezar, os produtos daquela bolsa serão descartados. Se o produto tiver a validade vencida também é desprezado. Depois de rotulado, o produto passa para uma outra câmara fria e lá fica armazenado até ser distribuído.

No setor de distribuição, os técnicos recebem as requisições de bolsas de sangue, verificam novamente a validade do produto antes de realizarem os testes de compatibilidade. Se a validade estiver em dia realizam os testes de compatibilidade e dão saída aos produtos aptos à transfusão. Caso contrário, o produto é desprezado. As saídas deverão ser registradas no sistema, bem como as devoluções e baixas de produtos.

A seguir será descrito o estudo de viabilidade para este estudo de caso.

6.4.2.2.O Estudo de Viabilidade do SADI

O estudo de viabilidade tem como objetivo propor soluções para a informatização das Unidades de Coleta e Transfusão da Fundação HEMOAM, e avaliar dentre as soluções apresentadas, aquela mais viável, levando em consideração critérios técnicos, operacionais,

financeiros e prazos requeridos. As informações apresentadas foram obtidas a partir de entrevistas aos funcionários da Fundação HEMOAM.

Para o levantamento de alternativas existentes em outros hemocentros, efetuamos pesquisas na Internet e consulta através de e-mail a funcionários do Datasus. Descobrimos que no Brasil existem quatro situações: hemocentros que utilizam o sistema de banco de sangue HEMOVIDA, outros que utilizam o sistema de banco de sangue da SBS, alguns hemocentros possuem sistema próprios e outros que ainda não estão informatizados.

A maioria dos hemocentros (Acre, Distrito Federal, Goiás, Maranhão, Mato Grosso, Paraíba, Piauí, Rio Grande do Sul, Sergipe, Tocantins, Paraná, e alguns hemocentros em São Paulo) nas capitais do país (46,3%), utilizam o sistema HEMOVIDA. O Sistema HEMOVIDA foi desenvolvido pelos técnicos do Datasus em Brasília, utiliza plataforma de banco de dados SQL Server.

Outros hemocentros (Alagoas, Ceará, Pará, Pernambuco, Santa Catarina, e alguns hemocentros em São Paulo) percentual de (20,4%) utilizam o sistema SBS. Desenvolvido pela SBS-Consultores em São Paulo, utiliza plataforma de banco de dados Progress.

Os hemocentros (Amazonas, Bahia, Minas Gerais, Rio de Janeiro, Rio Grande do Norte) possuem sistemas próprios; somam (18,5%) e finalmente os que ainda não estão informatizados (Amapá, Espírito Santo, Rondônia, Roraima) somam um percentual de (14,8%).

Apesar disso, os hemocentros que estão informatizados fornecem serviços para postos de coleta ou agências transfusionais não informatizados.

6.4.2.2.1.A situação atual no Estado do Amazonas

As Unidades de Coleta e Transfusão não estão informatizadas.

O acompanhamento das informações sobre doações e transfusões sanguíneas é feito manualmente, as informações são anotadas em livros.

Os exames para liberação da bolsa para transfusão são realizados na capital. Cada UCT (atualmente 47) manda a amostra de sangue e a ficha do doador com informações da doação para capital. As UCTs recebem os resultados dos exames sorológicos por telefone/fax para poder liberar a bolsa de sangue para transfusão.

A Fundação HEMOAM optou em utilizar, na capital, um sistema próprio (SAD) para o acompanhamento de suas informações sobre doações e transfusões sanguíneas, conforme dito anteriormente.

6.4.2.2.2.O problema identificado no Estado do Amazonas

Conforme descrito em tópicos anteriores deste trabalho, os problemas existentes são: identificação dos dados do doador ilegível (dados escritos manualmente); as fichas com as informações do doador chegam na capital incompletas sem uma série de dados importantes; o tempo de recebimento, dos exames realizados, pelo doador é longo, pois são enviados pelo correio. Dependendo da distância do município chega ser até dois meses.

6.4.2.2.3.Definição do escopo do sistema proposto

Esta definição foi descrita anteriormente no tópico “descrição do estudo de caso – sistema SADI”.

6.4.2.2.4.Alternativas identificadas

A *primeira* alternativa seria desenvolver o sistema para o interior do Estado, com a equipe de informática existente na Gerência de Sistemas na Fundação HEMOAM e propor a

alternativa de comunicação de dados. A *segunda* alternativa seria a aquisição de um sistema de terceiros (HEMOVIDA ou SBS).

6.4.2.2.5. Análise das alternativas identificadas

No processo de escolha da solução foram levantados dados técnicos de configuração, operacionalidade, possibilidade de integração, funcionalidade entre outros. Foram pesquisados todos os itens referente à funcionalidade ideal para o HEMOAM, atendendo os requisitos de software básico, e banco de dados padrão utilizado na instituição.

Foi criada uma estimativa de custo, levando em conta o custo total. Os valores para a implantação ou o desenvolvimento dependem de valores que podem variar com o tempo como: custo inicial de aquisição, passagens aéreas, hospedagem, transporte e até os salários dos desenvolvedores, no caso da adoção da solução proprietária.

Como os valores monetários podem sofrer variações com o tempo, foi calculado o custo total de cada um e, opinou-se por criar um esquema de proporcionalidade entre eles, onde cada um deles será uma variável que se diferencia pelo índice proporcional.

Devido os sistemas HEMOVIDA e SBS, terem matrizes localizadas em outros estados, será necessário incluir algumas despesas em cima do custo inicial do sistema como: passagens aéreas, hospedagens, transportes entre outras. Então como cada um deles tem um preço de aquisição específico, foram somadas a eles despesas, e criada uma variável que representa financeiramente cada solução, conforme equação abaixo:

Sistema HEMOVIDA = (CUSTO INICIAL + DESPESAS) => X

Sistema SBS = (CUSTO INICIAL + DESPESAS) => Y

O desenvolvimento de sistema SADI irá demandar, para um cronograma de um ano, honorários de três desenvolvedores. Calculando o salário-mês de cada um deles e multiplicando pelo número de meses no ano, chegou à proporção Z.

Sistema SADI = (CUSTO TOTAL) = Z

O valor do Sistema SAD que representa 3/4 do valor do sistema HEMOVIDA.

O valor do sistema SBS representa 100% a mais que o HEMOVIDA.

Concluindo-se, então que: Z =3X/4 e X=Y/2.

Tomando como base o variável X, o custo total do sistema SBS e o desenvolvimento próprio representam, respectivamente: 100% a mais que o custo X e 25% a menos que o custo de X. Desta forma concluímos que o sistema SADI é a solução mais barata, sendo 25% mais barato que adquirir o Sistema HEMOVIDA. O Sistema HEMOVIDA ficou em segundo lugar com relação a menor custo e por último está o Sistema SBS que é o dobro do valor do HEMOVIDA.

6.4.2.2.6.Vantagens e desvantagens das alternativas identificadas

Primeira Solução: desenvolver o sistema SADI.

Vantagens:

i. modelagem do sistema moldada aos processos estabelecidos e não os processos adequando-se ao sistema;
ii. utilização da mesma plataforma de banco de dados, facilitando a integração dos dados com o sistema da capital;
iii. utilização de uma linha telefônica convencional para acesso discado;
iv. facilidade de atendimento de manutenção quando o sistema tiver sido implantado, a equipe estará “in loco”, não necessitando pagar pelo deslocamento de técnicos de outros estados;
v. equipe de informática da Gerência de sistemas qualificada para desenvolvimento do sistema;

vi. não será necessário ter um contrato de manutenção do sistema depois de implantado, já que a equipe de informática pertence a Fundação HEMOAM;

vii. o código-fonte do sistema pertencerá a Fundação HEMOAM.

Desvantagens:

i. o tempo de desenvolvimento é maior que adquirir um sistema pronto. Mínimo 12 meses.

Segunda Solução: aquisição do sistema HEMOVIDA ou SBS

Vantagens:

i. o tempo para implantação de um dos sistemas é menor que o de desenvolver. Aproximadamente 6 meses;

ii. ambos sistemas apresentam boa funcionalidade;

iii. ambos possuem interface amigável;

Desvantagens:

i. impossibilidade de importação de dados históricos do doador (ambos), pois os sistema utilizado na capital é próprio;

ii. os processos atuais da Fundação HEMOAM terão que se adequar ao sistema (ambos);

iii. ambos sistemas não utilizam o mesmo SGBD adotado pelo HEMOAM, dificultando a integração dos dados com a capital;

iv. utilização de uma linha dedicada para troca de dados juntamente com uma rede VPN (Virtual Private Network);

v. o suporte poderá ser dificultado pela equipe não estar “in loco”, necessitando pagar pelo deslocamento de técnicos que virão de outro estado;

vi. será necessário ter um contrato de manutenção do sistema depois de implantado, já que a equipe que desenvolveu o sistema não pertence a Fundação HEMOAM;

vii. o código-fonte do sistema não pertencerá a Fundação HEMOAM.

6.4.2.2.7.Recomendação

A partir do estudo realizado concluímos pela adoção da primeira solução: desenvolvimento do sistema SADI**,** em virtude dos seguintes argumentos:

O número de vantagens em desenvolver é maior do que adquirir; devido as particularidades encontradas no levantamento de requisitos. A única desvantagem encontrada, foi o tempo de desenvolvimento ser maior do que o tempo de aquisição de um sistema pronto.

A aquisição de um dos sistemas apresentados (HEMOVIDA ou SBS) impossibilitaria a importação de dados históricos do doador, o que é de fundamental importância para Direção do hemocentro.

Os processos atuais da Fundação HEMOAM teriam que se adequar a ambos sistemas, enquanto que o desenvolvimento do SADI seria moldado a esse processo.

O sistema da capital, SAD, é uma solução proprietária que utiliza um SBGD diferente dos adotados em ambos sistemas (HEMOVIDA e SBS), dificultando a integração dos dados das unidades de coleta e transfusão do interior com a capital.

O alto custo de manutenção de um dos sistemas (HEMOVIDA ou SBS) devido à equipe de suporte não estar “in loco” não se justifica. Sendo necessário ter que pagar pelo deslocamento de técnicos de outro estado, sempre que necessário. Este é um critério de fundamental importância para adoção da primeira solução, pois o acesso às localidades do interior é difícil, devido a geografia do Estado Amazonas ser repleta de rios e florestas. É preciso diminuir ao máximo o risco de problemas tanto de hardware como de software, evitando o deslocamento de técnicos para essas localidades.

Os recursos financeiros para esse projeto são limitados, para adquirir qualquer um dos sistemas a Fundação HEMOAM teria que desembolsar um valor inicial maior do que a

quantia mensal que utilizaria para pagamento de salários para equipe de desenvolvimento, pois os recursos de hardware e software a fundação já possui.

Para a implantação de um dos sistemas (HEMOVIDA ou SBS) seria necessário utilizar um rede VPN (Virtual Private Network) e uma linha privada (LP). De acordo com a análise feita para aquisição desses requisitos, concluiu-se que não se justifica adotar esta solução, pois a demanda de dados, em algumas localidades no interior, é muito pequena.

Então recomendamos o desenvolvimento do sistema SADI para as unidades de coleta e transfusão no interior, pelas vantagens citadas e por ser uma solução que se encaixa dentro dos recursos financeiros disponíveis no momento e se adequa a utilização uma linha telefônica convencional para acesso discado.

6.4.2.3. Modelagem de Negócios do Sistema SADI

Após o levantamento de requisitos é necessário especificá-los. Na *modelagem de negócios* fazemos a modelagem dos requisitos do sistema para que estes fiquem registrados e possam ser validados. Um caso de uso descreve a seqüência de eventos de um ator (um agente externo) que usa o sistema para completar um processo. Eles não são exatamente a especificação de requisitos ou especificação funcional, mas ilustram e implicam requisitos na história que eles contam.

Vamos nos basear, para descrição dos casos de uso deste estudo de caso, no livro Desenvolvendo aplicações com UML, da autora Ana Cristina Melo (MELO, 2002), que dá um enfoque bem simples de como desenvolver a modelagem de um estudo de caso usando a UML. Nele são abordados o diagrama de casos de uso, a especificação do cenário principal, e a especificação do cenário alternativo. Ressaltamos novamente que existem outros estilos de descrever os casos de uso, mas nos limitaremos a adotar este enfoque.

O diagrama de caso de uso descreve um conjunto de seqüências de ações, cada uma representando a interação de itens externos ao sistema com o sistema. Cada caso de uso modela um requisito funcional do sistema. O cenário principal descreve uma seqüência de ações que serão executadas considerando que nada de errado ocorrerá durante a execução da seqüência. As exceções serão representadas nos cenários alternativos.

Os casos de uso representam conjuntos bem definidos de funcionalidades do sistema, que não podem trabalhar sozinhas no contexto do sistema. Portanto esses casos de uso precisam ser relacionar com outros casos de uso e com atores que enviarão e receberão mensagens destes.

Para relacionamentos de casos de uso, entre si, temos os tipos: *generalização*, *extensão* e *inclusão*. Para relacionamentos de atores, entre si, temos um único tipo, que é o relacionamento *generalização*. Para relacionamentos entre atores e casos de uso, temos apenas a *associação*.

Generalização ocorre entre casos de uso ou entre atores. É considerada quando temos dois elementos semelhantes, mas com um deles realizando algo mais.

Extensão entre casos de uso indica que um deles terá seu procedimento acrescido, em um ponto de extensão, de outro caso de uso, identificado como base.

Inclusão entre casos indica que um deles terá seu procedimento copiado num local específico no outro caso de uso, identificado como base.

Associação representa a interação do ator com o caso de uso, ou seja, a comunicação entre atores e casos de uso, por meio de envio e recebimento de mensagens. As associações entre casos de uso são sempre binárias, ou seja, envolvem apenas dois elementos. Representam o único relacionamento possível entre atores e casos de uso.

A partir do escopo do sistema, vamos identificar uma lista de casos de uso.

i. acessar sistema;

ii. consultar doador;

iii. cadastrar novo doador;

iv. consultar resultado sorologia;

v. cadastrar triagem;

vi. registrar inaptidão;

vii. cadastrar doação;

viii.registrar reação;

ix. cadastrar imunohematologia;

x. enviar dados para capital;

xi. receber dados da capital;

xii. cadastrar geração de produtos;

xiii.registrar intercorrência;

xiv.consultar conduta da bolsa;

xv. rotular produto gerado;

xvi.cadastrar distribuição do produto;

xvii.cadastrar devolução do produto;

xviii.cadastrar descarte por validade.

A partir da lista de casos de uso, vamos agora criar os casos de uso e descrever sua funcionalidade através dos cenários.

6.4.2.3.1.Casos de Uso: acessar sistema

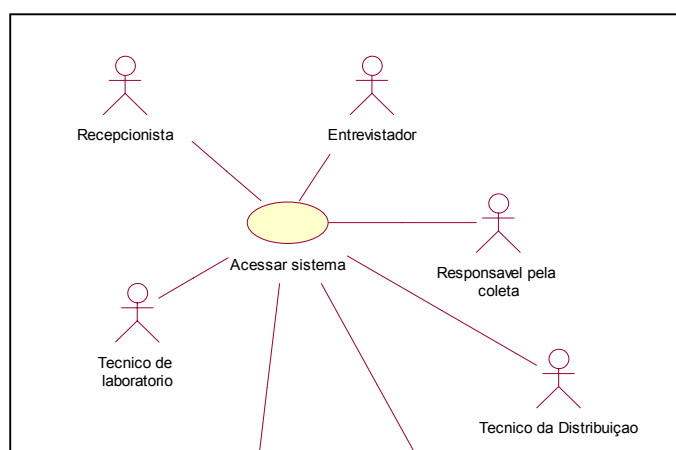



Figura 27 – Caso de uso: acessar sistema

A Figura 27 mostra o diagrama de caso de uso *acessar sistema* que é a representação gráfica dos atores e cenários principal e alternativo a seguir.

Atores: recepcionista, entrevistador, responsável pela coleta, técnico de laboratório, técnico do fracionamento, técnico da rotulagem, técnico da distribuição.

Cenário Principal

1. Qualquer um dos atores, "usuários", acessam o sistema, pelo nome e senha.

2. Login OK. O Sistema mostra a tela principal para que o ator escolha a opção.

Cenário Alternativo - O usuário não possui acesso ao sistema

2A. Login negado. O Sistema não dá acesso ao usuário utilizar o sistema.

Os atores possuem um relacionamento de associação com o caso de uso acessar sistema.

Nos cenários alternativos, as alternativas numéricas seguidas de uma letra são apresentadas como subitens do cenário principal (2-2A como mostra os cenários do caso de uso anterior). Esse procedimento facilita a associação da alternativa com o fluxo principal.

6.4.2.3.2.Casos de Uso: consultar doador e cadastrar novo doador

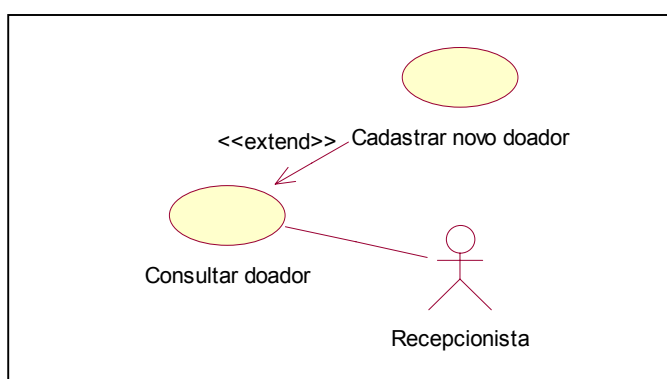



Figura 28 – Casos de uso: consultar doador e cadastrar novo doador

A Figura 28 mostra o ator *recepcionista* associado ao diagrama de caso de uso *consultar doador* que é o caso de uso base e possui seu procedimento acrescido através de uma extensão do caso de uso *cadastrar novo doador*. Neste caso o processo de consulta é extendido para um cadastro quando em sua pesquisa é descoberto que um determinado doador ainda não possui cadastro, porque para se cadastrar um novo doador obrigatoriamente é necessário uma consulta anterior.

Extensão entre casos de uso indica que um deles terá seu procedimento acrescido, em um ponto de extensão, de outro caso de uso, identificado como base.

A seguir temos os cenários principal e alternativo do caso de uso *consultar doador*.

Ator: recepcionista

Cenário Principal

1. O recepcionista verifica no Sistema, pelo nome e data do nascimento, se o doador já possui cadastro.

2. O Sistema retorna uma lista com nomes e data do nascimento.

3. O recepcionista seleciona nome da lista.

4. O Sistema mostra dados e doador apto.

5. O recepcionista solicita impressão da ficha.

Cenário Alternativo

Doador não cadastrado

1A. O recepcionista verifica no Sistema, pelo nome e data do nascimento, se o doador já possui cadastro. Caso negativo, o usuário cadastra novo doador.

Doador inapto

4A. O Sistema informa que o doador está inapto. A inaptidão pode ser por prazo insuficiente ou por problemas na doação anterior.

A seguir temos o cenário principal do caso de uso *cadastrar novo doador*.

Ator: recepcionista

<u>Cenário Principal</u>

1. O recepcionista verifica no Sistema, pelo nome e data do nascimento, se o doador já possui cadastro.

2. O Sistema retorna uma lista com nomes. O nome não está na lista.

3. O recepcionista realiza cadastro do doador, informando os dados: nome, data do nascimento, naturalidade, nacionalidade, estado civil, sexo, identidade, cpf, nome do pai, nome da mãe, endereço, bairro, cep, cidade, estado, telefone, data de cadastro.

5. O recepcionista solicita impressão da ficha.

6.4.2.3.3.Caso de Uso: consultar resultado sorologia

A Figura 29 mostra o ator *recepcionista* associado ao diagrama de caso de uso *consultar resultado de sorologia*. A seguir temos os cenários principal e alternativo deste caso de uso.

Ator: recepcionista

<u>Cenário Principal</u>

1. O recepcionista verifica no Sistema, pelo nome e data do nascimento, se o doador já possui resultado.

2. O Sistema retorna uma lista com nomes e data do nascimento.

3. O recepcionista seleciona nome da lista.

4. O Sistema mostra dados exame.

5. O recepcionista solicita impressão dos exames.

Cenário Alternativo - Doador sem resultado

1A. O recepcionista verifica no Sistema, pelo nome e data do nascimento, se o doador já possui resultado. Caso negativo, o recepcionista informa ao doador que seus resultados ainda não chegaram da capital.

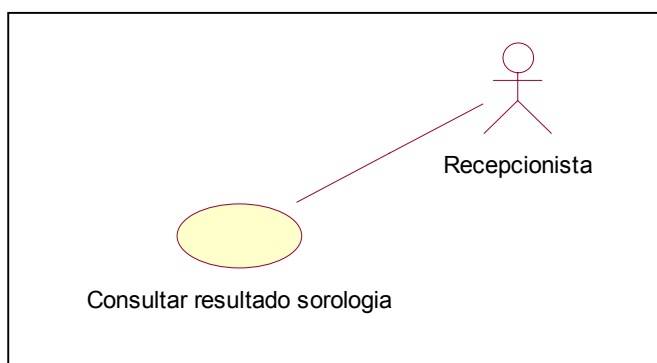



Figura 29 – Caso de uso: consultar resultado sorologia

6.4.2.3.4.Casos de Uso: cadastrar triagem e registrar inaptidão

A Figura 30 mostra o ator *entrevistador* associado ao diagrama de caso de uso *cadastrar triagem* que é o caso de uso base e possui seu procedimento acrescido através de uma extensão do caso de uso *registrar inaptidão*. A seguir temos os cenários principal e alternativo do caso de uso *cadastrar triagem*. A extensão aqui se justifica pelo fato de que um registro de inaptidão só poder ser realizado quando uma triagem é cadastrada.

Ator: Entrevistador

Cenário Principal

1. O entrevistador realiza cadastro da triagem, no Sistema, informando os dados: peso, altura, pressão arterial, hematócrito, temperatura, data triagem, procedência, observações, apto para doar, tipo da doação, resultado triagem.

Cenário Alternativo - Doador Inapto

1A. O entrevistador registra inaptidão, no Sistema.

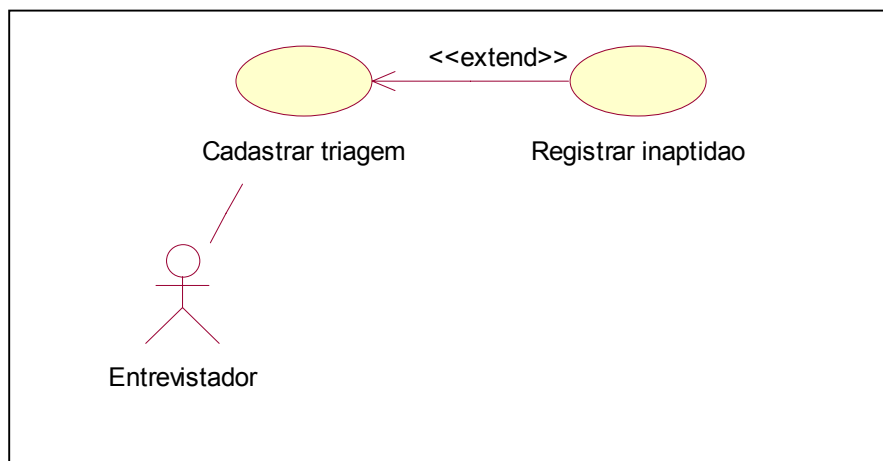



Figura 30 – Casos de uso: cadastrar triagem e registrar inaptidão

A seguir temos o cenário principal do caso de uso *registrar inaptidão*.

Ator: Entrevistador

Cenário Principal

1. O entrevistador registra inaptidão, no Sistema.

6.4.2.3.5.Casos de Uso: cadastrar doação e registrar reação

A Figura 31 mostra o ator *responsável pela coleta* associado ao diagrama de caso de uso *cadastrar doação* que é o caso de uso base e possui seu procedimento acrescido através de uma extensão do caso de uso *registrar reação*. A extensão, também se justifica neste caso, pois uma reação só poderá ser registrada quando inicialmente uma doação for cadastrada.

A seguir temos os cenários principal e alternativo do caso de uso *cadastrar doação*.

Ator: Responsável pela Coleta

Cenário Principal

1. O responsável pela coleta realiza cadastro da doação, no Sistema, informando os dados: numero da bolsa, data coleta, hora coleta, volume doado, local de coleta, interior/capital.

Cenário Alternativo - O doador teve reação na coleta

1A. O responsável pela coleta registra reação, no Sistema.

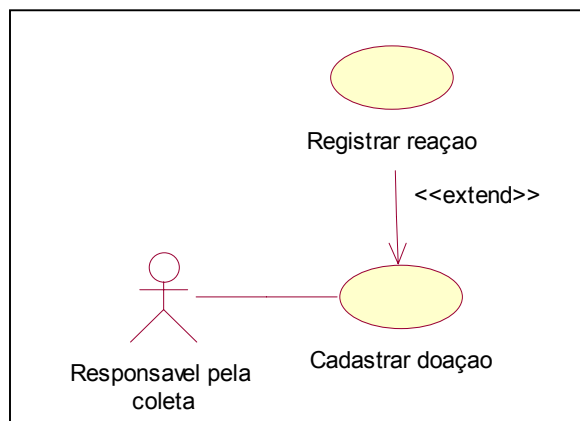



Figura 31 – Casos de uso: cadastrar doação e registrar reação

A seguir temos o cenário principal do caso de uso *registrar reação*.

Ator: Responsável pela coleta

Cenário Principal

1. O responsável pela coleta registra reação, no Sistema.

6.4.2.3.6.Caso de Uso: cadastrar imunohematologia

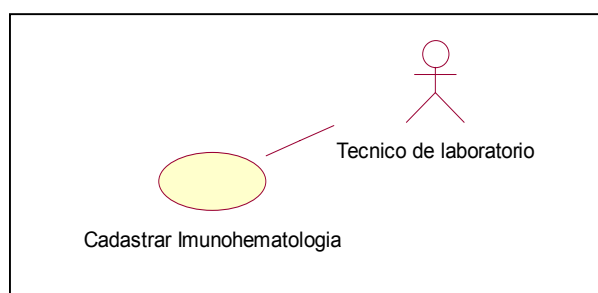



Figura 32 – Caso de uso: cadastrar imunohematologia

A Figura 32 mostra o ator *técnico de laboratório* associado ao diagrama de caso de uso *cadastrar imunohematologia*. A seguir temos o cenário principal deste caso de uso.

Ator: Técnico de laboratório

Cenário Principal

1. O Técnico de laboratório realiza cadastro da imunologia, no Sistema, informando os dados: tipo sanguineo, fator Rh, pesquisa de anticorpos, data imuno, hora imuno, conduta imuno.

6.4.2.3.7.Casos de Uso: enviar dados para capital e receber dados da capital

A Figura 33 mostra o ator *técnico de laboratório* associado a dois diagramas de caso de uso *enviar dados para capital* e *receber dados da capital*. A seguir temos o cenário principal do caso de uso *enviar dados para capital*.

Ator: Técnico de laboratório

<u>Cenário Principal</u>

1. O Técnico de laboratório informa o período: data inicio, data final

2. O Técnico de laboratório solicita, no Sistema, a geração do arquivo de exportação contendo os dados: novos doadores cadastrados, triagem, doação, imunohematologia.

3. O Sistema gera o arquivo e coloca-o na pasta arquivos gerados.

4. O Técnico de laboratório envia o arquivo por e-mail para capital.

A seguir temos o cenário principal do caso de uso *receber dados da capital*.

Ator: Técnico de laboratório

<u>Cenário Principal</u>

1. O Técnico de laboratório recebe por e-mail o arquivo da capital.

2. O Técnico de laboratório copia o arquivo para a pasta arquivos recebidos

3. O Técnico de laboratório solicita, no Sistema, a importação dos dados do arquivo para base local.

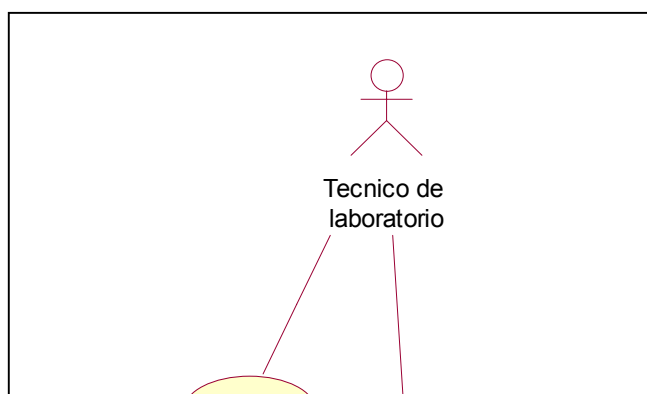



Figura 33 – Casos de uso: enviar dados para capital e receber dados da capital

6.4.2.3.8.Casos de Uso: cadastrar geração de produtos e registrar intercorrência

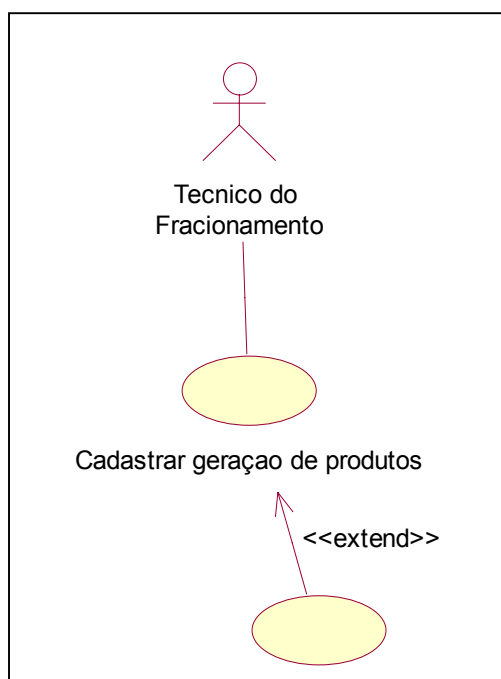



Figura 34 – Casos de uso: cadastrar geração de produtos e registrar intercorrência

A Figura 34 mostra o ator *técnico do fracionamento* associado ao diagrama de caso de uso *cadastrar geração de produtos* que é o caso de uso base e possui seu procedimento acrescido através de uma extensão do caso de uso *registrar intercorrência*. A seguir temos os cenários principal e alternativo do caso de uso *cadastrar geração de produtos*. A extensão neste caso se justifica pelo fato de que um registro de uma intercorrência só poderá ser realizado após o cadastro da geração de um produto.

Ator: Técnico do Fracionamento

Cenário Principal

1. O Técnico do Fracionamento informa o número da doação e o código produto. Nenhuma Intercorrência

2. O sistema verifica data da coleta.

3. O Técnico do Fracionamento realiza cadastro dos produtos gerados, no Sistema, informando os dados: data geração, hora geração, status do produto=gerado, data vencimento=data geração+validade.

Cenário Alternativo - Intercorrência

3A. O Técnico de Fracionamento registra intercorrência, no Sistema

A seguir temos o cenário principal do caso de uso *registrar intercorrência*.

Ator: Técnico do Fracionamento

Cenário Principal

1. O Técnico do Fracionamento registra no Sistema intercorrência e status produto=descarte intercorrencia.

6.4.2.3.9. Casos de Uso: consultar conduta da bolsa e rotular produto gerado

A Figura 35 mostra o ator *técnico da rotulagem* associado a dois diagramas de caso de uso *consultar conduta da bolsa* e *rotular produto gerado*. A seguir temos os cenários principal e alternativo do caso de uso *consultar conduta da bolsa*.

Figura 35 – Casos de uso: consultar conduta da bolsa e rotular produto gerado

Ator: Técnico da Rotulagem

Cenário Principal

1.O Técnico da Rotulagem informa numero da bolsa

2. O Sistema verifica doação e lista conduta. Tem conduta. Mostra conduta liberada.

Cenário Alternativo

Tem conduta e conduta desprezar

2A. Tem conduta. O Sistema mostra conduta desprezar

Não tem conduta

2A. Não tem conduta. O Sistema mostra sem conduta

A seguir temos os cenários principal e alternativo do caso de uso *rotular produto gerado*.

Ator: Técnico da Rotulagem

Cenário Principal

1. O Técnico da Rotulagem informa numero bolsa, codigo produto.

2. O sistema verifica validade do produto. Validade ok.

3. O sistema verifica conduta da bolsa. Conduta liberada.

4. A etiqueta é impressa.

5. O sistema cadastra status do produto=rotulado, data rotulagem, hora rotulagem.

Cenário Alternativo

Validade não ok..

2A. O sistema verifica validade do produto. Validade não ok.

4A. A etiqueta não é impressa.

5A. O sistema cadastra status do produto=descarte por vencimento.

Bolsa com conduta desprezar.

2A. O sistema verifica conduta da bolsa. Conduta desprezar.

4A. A etiqueta não é impressa.

5A. O sistema cadastra status do produto=descarte por exame.

6.4.2.3.10.Casos de Uso: cadastrar distribuição do produto e cadastrar devolução do produto

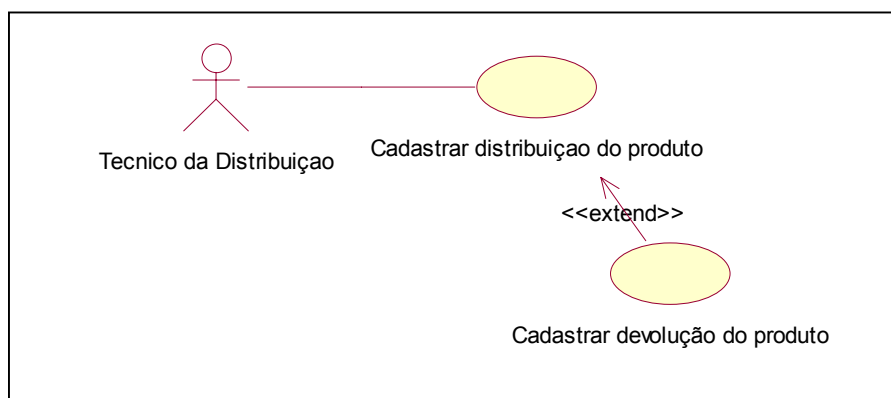



Figura 36 – Caso de uso: cadastrar distribuição do produto e cadastrar devolução do produto.

A Figura 36 mostra o ator *técnico da distribuição* associado ao diagrama de caso de uso *cadastrar distribuição do produto* que é o caso de uso base e possui seu procedimento acrescido através de uma extensão do caso de uso *cadastrar devolução do produto*. A seguir temos os cenários principal e alternativo do caso de uso *cadastrar distribuição do produto*. Novamente a extensão neste caso é que para realizar o cadastro de uma devolução do produto é necessário antes cadastrar a distribuição do mesmo.

Ator: Técnico da Distribuição

Cenário Principal

1. O Técnico da distribuição informa (numero bolsa, código produto).

2. O sistema verifica doação e produto. OK

3. O sistema realiza cadastro: data distribuição, hora distribuição, status produto=distribuido.

Cenário Alternativo - Doação ou produto não existentes

2A. O sistema verifica doação e produto. Não OK.

3A. O sistema não realiza o cadastro.

A seguir temos o cenário principal do caso de uso *cadastrar devolução do produto*.

Ator: Técnico da Distribuição

Cenário Principal

1. O Técnico da distribuição informa (numero bolsa, codigo produto).

2. O sistema realiza cadastro: data retorno, hora retorno, status produto=retornado.

6.4.2.3.11.Caso de Uso: cadastrar descarte por validade

A Figura 37 mostra o ator *técnico da distribuição* associado ao diagrama de caso de uso *cadastrar descarte por validade*. A seguir temos o cenário principal deste caso de uso.

Ator: Técnico da Distribuição

Cenário Principal

1. O Técnico da distribuição informa (numero bolsa, codigo produto).

2. O sistema realiza cadastro: data descarte, hora descarte, status produto=descartado por vencimento.

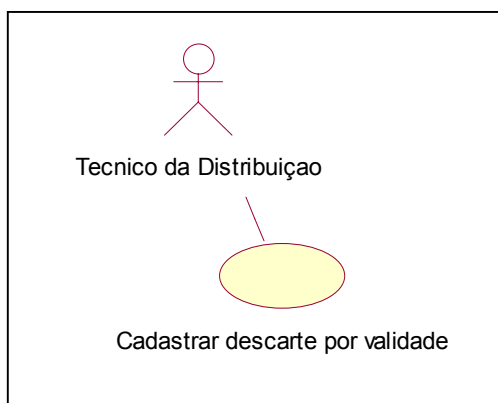



Figura 37 – Caso de uso: cadastrar descarte por validade.

6.4.2.4.Modelagem do Domínio do Problema do Sistema SADI

Na modelagem do domínio do problema identificaremos as classes do domínio do problema e seus atributos. Vamos elaborar o diagrama de classes para modelar detalhes das classes e seus relacionamentos.

A partir dos casos de uso, vamos analisar um a um e identificar classes e atributos, depois, otimizaremos o modelo criando relacionamentos: associação, generalização, dependência, etc.

A *associação* é um relacionamento que conecta duas ou mais classes, demonstrando a colaboração entre as instâncias de classe. A associação possui um nome que é mostrado próximo a linha do relacionamento. Além do nome a associação também possui a multiplicidade que é colocada nas extremidades do caminho da associação identificando o número de instâncias de uma classe que pode se relacionar com a outra.

A *generalização* entre classes quando uma classe possui subclasses, as quais herdam os atributos e serviços da classe mãe.

O relacionamento de *dependência* entre duas classes indica que uma mudança na interface de uma delas pode causar mudança na outra.

A partir desses breves conceitos vamos mostrar o diagrama de classes criado e fazer uma breve explicação do mesmo.

6.4.2.4.1.Diagrama de Classes

Foram identificadas as seguintes classes: doador; triagem; motivo inaptidão; doação; exame sorologia; imunohematologia; resultado exame sorologia; produto; produto gerado; status produto.

Conforme mostra a Figura 38, a *classe doador* tem um relacionamento de associação com a *classe triagem*. Um doador de sangue pode ter nenhuma ou várias triagens. A *classe triagem* tem um relacionamento de associação com a *classe motivo inaptidão*. As triagens de um doador podem estar associadas a nenhum ou um motivo de inaptidão. A *classe triagem* também possui um relacionamento de associação com a *classe doação*. Uma triagem pode estar associada a nenhuma ou uma doação. A *classe doação* também possui um relacionamento de associação "um-pra-um" com a *classe imunohematologia*. Cada doação está associada a uma imunohematologia. Existe um relacionamento ternário de associação entre as classes: *doação*; *exame sorologia*; *resultado exame sorologia*. Isso indica que para cada doação pode haver vários resultados de exames sorológicos, onde cada resultado está associado a um exame sorológico. Da mesma forma, há um relacionamento ternário de associação entre as classes: *doação*; *produto* e *produto gerado*. Indicando que para cada doação pode haver vários produtos gerados a partir daquela bolsa de sangue, e esses produtos estão associados a uma tabela de produtos. A *classe produto gerado* possui um relacionamento de dependência com a *classe status do produto*. As operações dessas classes encontradas serão descobertas após a criação dos diagramas de seqüência, na fase de projeto.

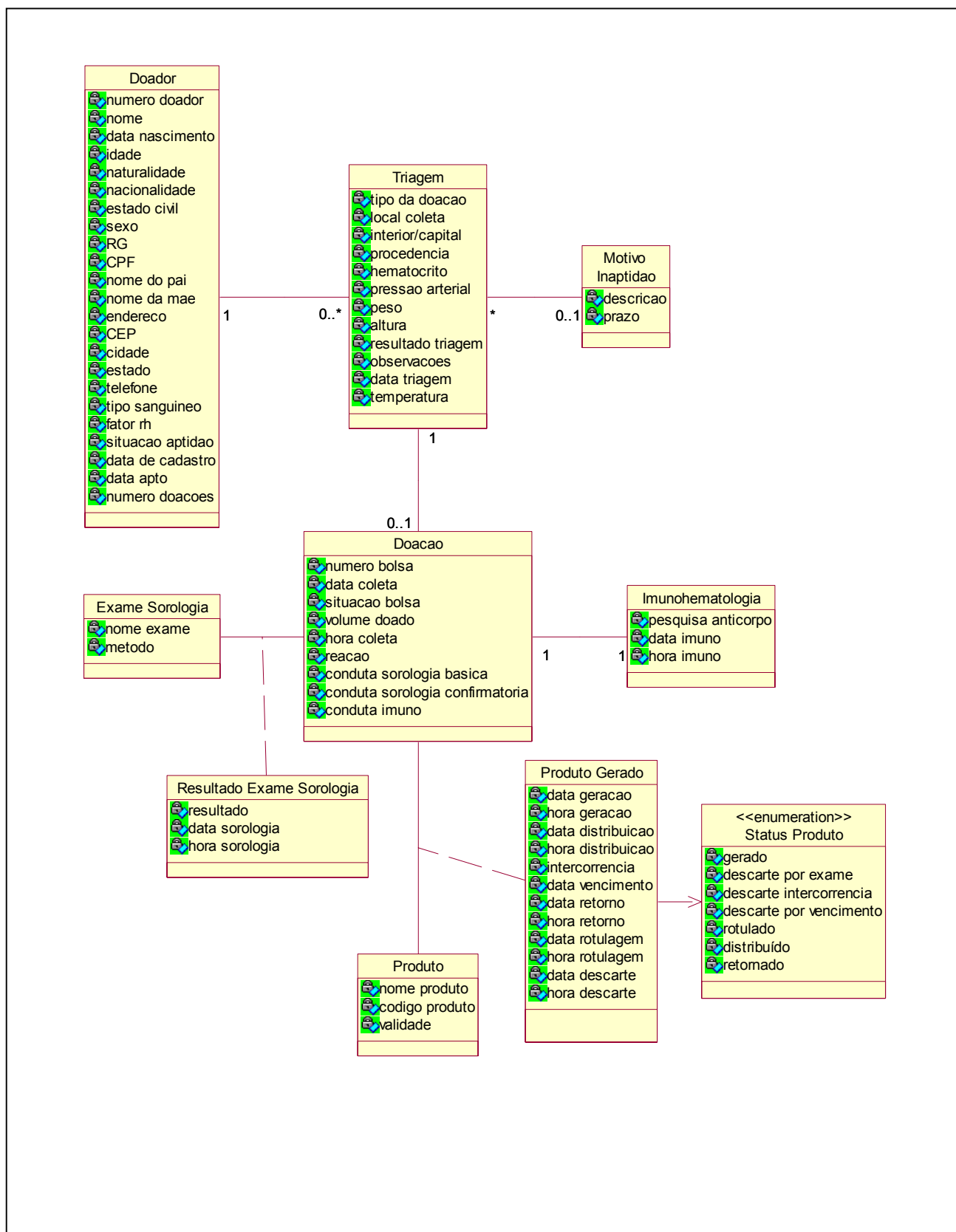



Figura 38 – Diagrama de Classes

As classes de interface não serão representadas nos diagramas de classe, deste trabalho, nos preocupamos apenas em representar as classes do domínio do problema.

A seguir descreveremos os atributos de cada classe.

6.4.2.4.1.1.Descrição dos atributos das Classes

A *classe doador* possui os seguintes atributos:

i. número doador – todo o doador adquire um número de registro. Esse número é utilizado nos processos internos, como também pode servir para agilizar o atendimento na recepção de doadores;
ii. nome completo*;*
iii. data de nascimento*;*
iv. idade*;*
v. naturalidade*;*
vi. nacionalidade*;*
vii. estado civil – a situação civil de um doador será identificada por letras correspondentes, ou seja, Casado – C, Solteiro – S, D- Divorciado;
viii. sexo – necessário apenas cadastrar a primeira letra das alternativas, por exemplo, a letra “M” significa masculino e a letra “F” feminino;
ix. RG - número da carteira de identidade;
x. CPF- número do CPF;
xi. nome completo do pai;
xii. nome completo da mãe;
xiii. endereço residencial completo – descrição do logradouro e número, apartamento, Descrição do Bairro etc;
xiv. CEP- número do CEP;

xv. cidade em que reside;

xvi.estado em que reside;

xvii.telefone para Contato

xviii.tipo Sanguíneo

xix.RH – fator Rh

xx. situação de aptidão – esse dado é importante, pois a partir dele é feita a liberação ou não de um nova doação para o candidato;

xxi.data de cadastro – data que o doador foi cadastrado no sistema;

xxii.data apto – caso o doador tenha ficado inapto, por algum motivo, este campo será calculado e mostrado na interface. O campo indica a data em que o doador estará apto a doar novamente;

xxiii.numero doações – este campo também é calculado e mostrado na interface. Indica a quantidade de doações feitas pelo doador.

A *classe triagem* representa os dados da entrevista do doador, possui os seguintes atributos:

i. tipo da doação – sigla que referencia o tipo de doação escolhida. Pode ser: Voluntária(V), Pré-depósito(P) ou Reposição(R);

ii. local de coleta – nome do local onde é realizada a coleta;

iii. interiorcapital – pode ser: Interior (I) ou Capital (C);

iv. procedência – última cidade em que doador transitou nos últimos meses;

v. hematócrito - resultado do exame hematócrito para verificar se a pessoa está com anemia;

vi. pressão arterial – medição de pressão arterial;

vii. peso;

viii. altura;

ix. resultado da triagem (aptidão)- S/N;

x. eventuais **observações** feitas pelo médico sobre o candidato a doação;

xi. data da entrevista – data em que o candidato a doação foi entrevistado pelo entrevistador(Médico);.

xii. temperatura.

A *classe doação* representa a coleta de sangue; que é autorizada quando o candidato a doação é aprovado na triagem. Depois que o sangue é coletado em uma bolsa específica, ele recebe um número de identificação que será necessário para o restante do processo. Possui os seguintes atributos:

i. número da bolsa – número que identifica unicamente a bolsa do doador;

ii. data da coleta - data em que o sangue é coletado;

iii. situação da bolsa - situação em que se encontra a bolsa de sangue depois da análise feita nos laboratórios. Se ela foi liberada para a estocagem ou se ela foi descartada;

iv. volume doado – é o volume de sangue coletado, geralmente cerca de 400 ml. Desde total serão tirados 3 tubinhos para as análises laboratoriais;

v. hora da coleta – hora que o sangue é coletado;

vi. reação – se houve reação do doador na coleta. Como desmaio, tontura, etc. A reação pode se dar durante a coleta ou após a coleta. Existem casos de reação durante a coleta em que o volume coletado de sangue não é suficiente para gerar produtos; a bolsa é descartada;

vii. conduta sorologia basica – liberar ou desprezar. A conduta básica libera a bolsa de sangue para transfusão, caso todos os resultados de exames da sorologia

básica forem “não reativos”. Se um dos exames da sorologia básica for “reativo” a bolsa de sangue é desprezada;.

viii. conduta sorologia confirmatória – doador apto ou inapto. A conduta da sorologia confirmatória libera ou não o doador para novas doações. Se o resultado do exame da sorologia confirmatória for “reativo”, a conduta confirmatória será “Doador Inapto”. Se o exame for “não reativo” o doador receberá a conduta “Doador Apto”;

ix. conduta imuno – a conduta da imunohematologia é “Liberar” ou “Desprezar” bolsa de sangue para transfusão. Quando a pesquisa de anticorpos tem resultado positivo, a bolsa é desprezada. Caso contrário sua conduta será Liberar.

Após o sangue ser coletado, ele é examinado em dois laboratórios específicos, Sorologia e Imunohematologia. Eles analisam o sangue através de onze exames regulamentados pelo ministério da saúde. Caso o sangue seja aprovado nessa análise, ele recebe condições positivas para ser estocado e distribuído para transfusão.

A *classe resultado exame sorologia* possui os seguintes atributos:

i. resultado do exame – o resultado depende do exame e método. Na maioria dos casos é Reativo ou Não reativo;
ii. data da sorologia – data de realização do exame sorológico;
iii. hora da sorologia – hora que foi realizado exame sorológico.

A *classe exame sorologia* possui os seguintes atributos:

i. exame – nome do exame realizado, por exemplo hepatite;

ii. metodo – é o método de realização do exame. Cada exame pode ter um ou mais métodos para sua realização. Esses métodos identificam a precisão do resultado. Por exemplo, o laboratório de sorologia realiza duas baterias de exame, básica e confirmatória. A sorologia básica para primeira análise do sangue e a sorologia confirmatória, caso o exame apresente o resultado soropositivo, para verificar o resultado obtido na sorologia básica.

A c*lasse imunohematologia* possui os seguintes atributos:

i. pesquisa de anticorpos – a pesquisa de anticorpos é um exame realizado para se detectar se o doador possui anticorpos que comprometam a transfusão. Seu resultado é Positivo ou Negativo;
ii. data imuno – data de realização da imunohematologia;
iii. hora imuno – hora de realização da imunohematologia.

A *classe produto gerado* possui os seguintes atributos:

i. data geração – data em que o produto foi gerado;
ii. hora geração – hora em que o produto foi gerado;
iii. data distribuição – data que o produto foi distribuído;
iv. hora distribuição – hora que o produto foi distribuído;
v. intercorrência – problema que impeça a geração do produto;
vi. status – status do produto: gerado, descartado, rotulado, distribuído ou retornado;
vii. data vencimento – data de vencimento do produto;
viii. data retorno – caso o produto, depois de distribuído, retorne ao HEMOAM sua data de retorno será registrada;

ix. hora retorno – caso o produto, depois de distribuído, retorne ao HEMOAm sua hora de retorno será registrada;

x. data rotulagem – quando o produto é rotulado, o sistema guarda data da rotulagem;

xi. hora rotulagem – quando o produto é rotulado, o sistema guarda hora da rotulagem;

xii. data descarte – se por algum motivo o produto for descartado sua data de descarte será registrada;

xiii. hora descarte - se por algum motivo o produto for descartado sua hora de descarte será registrada.

Esta classe guarda os produtos gerados a partir das bolsas de sangue coletadas, que possuem as condutas Liberadas (conduta imunohematologia e conduta sorologia básica)

A *classe produto* possui os seguintes atributos:

i. código produto – código do produto;

ii. nome do produto – nome do produto gerado. Concentrado de plaquetas, plasma, plaquetas, etc;

iii. validade – validade do produto.

A *classe motivo inaptidão* possui os seguintes atributos:

i. descrição – guarda a descrição dos motivos de inaptidão;

ii. prazo – prazo referente ao motivo de inaptidão em que o doador deverá permanecer sem doar.

A *classe status produto* guarda os “status” que um produto pode ter, como: gerado, descarte por exame, descarte por intercorrência, descarte por vencimento, rotulado, distribuído, retornado.

A seguir veremos as decisões de projeto do sistema.

6.4.2.5.Decisões do Projeto do sistema SADI

Nesta seção serão definidos o *tipo de banco de dados*, a *linguagem de programação* e a *interface com o usuário*, que serão utilizados na etapa de implementação. Vamos apenas descrever sucintamente estes três assuntos, pois o escopo deste trabalho, conforme já dissemos, vai até o final da etapa de projeto.

6.4.2.5.1.Modelagem em Banco Relacional

A aplicação SADI requer o armazenamento e a recuperação de informações em um mecanismo de armazenamento persistente que será um banco de dados relacional. Por causa da predominância atual dos gerenciadores de banco de dados relacional a Fundação HEMOAM adquiriu o Sistema Gerenciador de Banco de Dados Oracle há algum tempo atrás.

A performance e segurança fornecida por esse banco relacional nos leva a pensar em implementar o sistema nele. É claro que, a equipe de desenvolvimento está ciente do desencontro entre as representações de dados orientadas a registros e as representações orientadas a objetos.

Não existem mecanismos nativos nos bancos relacionais que permitam a implementação de conceitos próprios da orientação a objetos, como: herança e polimorfismo. Nos acostumamos a enxergar a ligação entre tabelas dos modelos relacionais por meio de chaves primárias e secundárias. Já a ligação entre objetos é feita por referência, visto eles

possuírem uma identidade única. Para conseguirmos, então, que os objetos sejam persistidos em bancos relacionais, necessitamos fazer com que tabelas representem os objetos.

Para conseguirmos mapear os modelos de objetos em tabelas relacionais teremos que atender algumas necessidades, descritas a seguir:

i. cada classe é mapeada diretamente em uma ou mais tabelas do banco de dados. Para cada tabela derivada devemos ter uma chave primária. Alguns atributos podem se tornar chaves primárias desde que cumpram os pré-requisitos necessários para serem uma chave primária. Caso contrário, devemos criar um identificador para exercer tal papel;
ii. quanto aos relacionamentos de associação, podemos ter ou não o mapeamento de uma tabela. A exceção é a associação que possua multiplicidade de muitos para muitos. Esse caso sempre gerará uma tabela. O mapeamento da agregação segue as mesmas regras da associação.

6.4.2.5.2.Linguagem de programação

Segundo Deitel (2003):

> A linguagem **Java** é orientada a objetos, com forte suporte para técnicas adequadas de engenharia de software. Em Java, a unidade de programação é *classe*, a partir da qual, em algum momento os objetos são *instanciados* (“criados”). As classes de Java contêm *métodos* (que implementam os comportamentos da classe) e *atributos* (que implementam os dados da classe).

Os programadores de Java se concentram em criar seus próprios *tipos definidos pelo usuário*, chamados *classes* e *componentes*. Cada classe contém dados e o conjunto de funções que manipulam aqueles dados. Os componentes de dados de uma classe são conhecidos como *atributos*. Os componentes funções de uma classe são conhecidos como *métodos*. Da mesma maneira que uma instância de um tipo primitivo da linguagem, como **int**, é chamada *variável*, uma instância de um tipo definido pelo usuário (i.e., uma classe) é chamada de *objeto*. O programador usa tipos primitivos como blocos de construção para construir tipos definidos

pelo usuário. O foco de atenção em Java está nas classes (com as quais criamos objetos) e não nas funções. Os *substantivos* em uma especificação de sistemas ajudam o programador de Java a determinar o conjunto de classes a partir das quais serão criados objetos que irão trabalhar juntos para implementar o sistema.

Java não suporta herança múltipla, ela oferece a maioria dos benefícios fundamentais desta tecnologia através do suporte a múltiplas interfaces para uma classe.

Por ser uma linguagem de programação de uso geral, com recursos suficientes para a construção de uma variedade de aplicativos, Java se tornou uma linguagem completa para construção de aplicativos que podem ou não depender do uso de recursos de conectividade.

Java é uma linguagem simples, de fácil aprendizado ou migração, pois possui um reduzido número de construções. A diminuição das construções mais suscetíveis a erros de programação, tais como ponteiros e gerenciamento de memória via código de programação também faz com que a programação em Java seja mais segura. Contém um conjunto de bibliotecas que fornecem grande parte da funcionalidade básica da linguagem, incluindo rotinas de acesso à rede e criação de interface gráfica. Permite a modularização das aplicações, reuso e manutenção simples do código já implementado.

Os programas escritos em Java podem assumir duas formas básicas: *applets* e *applications*. A programação desses dois tipos têm características bem distintas. Segurança, interface gráfica, acesso à unidades de disco e acesso à rede são pontos de divergência entre os tipos. Entretanto, a diferença básica está no fato de que *applets* precisam de um *web browser* para existir. Isso quer dizer que *applets* são aplicações voltadas para o ambiente Internet/Intranet e que são transportadas pela rede junto com hiperdocumentos HTML. Já as *applications* são programas aplicativos, *stand-alone*, escritos para operar sem a necessidade de um *web browser* para montar sua interface gráfica; não possuem restrições de acesso a unidades de disco e não têm restrição de acesso à rede.

Diante de todos os benefícios apresentados, é esta a linguagem pensada para etapa de implementação.

6.4.2.5.3.Interface com o usuário

Para cada rotina desejada, será necessária uma tela para intermediar o acesso do usuário às classes de nossa aplicação. O modelo de casos de uso determina o projeto de interface. Ao reunir todos os casos de uso de nosso modelo, temos em mãos material suficiente para criar a estrutura de nossa aplicação.

A partir da lista de casos de uso, da análise de requisitos chegamos a seguinte estrutura de menu para o sistema SADI:

RECEPÇÃO

- Consultar
  - Doador
  - Resultado sorologia
- Cadastrar doador

TRIAGEM

- Cadastrar Triagem
  - Registrar Inaptidão

COLETA

- Cadastrar doação
  - Registrar reação

IMUNOLOGIA

- Cadastrar imunologia
- Dados
  - Exportar

▪Importar

FRACIONAMENTO

•Cadastrar geração de produtos

▪Registrar intercorrência

ROTULAGEM

•Rotular produto gerado

▪Consultar conduta bolsa

DISTRIBUIÇÃO

•Cadastrar distribuição produto

▪Cadastrar retorno produto

•Cadastrar descarte por validade

Os casos de uso se transformaram em itens do menu. Cada item de menu chamará uma tela, que pode pertencer à própria aplicação ou à classe. Se a tela pertencer à própria aplicação, ela será uma classe visual que se encarregará de coletar as informações do usuário e transmitir às classes, por meio de troca de mensagens.

Na próxima seção vamos começar a modelagem comportamental do sistema SADI.

6.4.2.6.Modelagem Comportamental do Sistema SADI

A modelagem comportamental descreve o comportamento do sistema, mostrando como são realizadas as interações entre os objetos, quais os estados nos quais um objeto pode estar e quais operações ele pode realizar. Para operações complexas, são mostradas as ações e atividades que as compõem. Os diagramas elaborados são diagrama de interação, diagramas de classe, diagramas de estado e diagrama de atividades.

Um *diagrama de interação* mostra uma interação formada por um conjunto de objetos e seus relacionamentos, incluindo as mensagens que poderão ser trocadas entre eles. Os diagramas de interação podem ser usados para descrever como os casos de uso são realizados como interações entre associações de objetos. Há dois tipos de diagramas de interação: diagrama de seqüência (que dá ênfase à ordenação temporal de mensagens) e o diagrama de colaboração (que dá ênfase à organização estrutural dos objetos que enviam e recebem as mensagens). Esses diagramas são semanticamente equivalentes, por isso vamos utilizar somente o diagrama de seqüência.

Os *diagramas de classe* são completados com operações descobertas através dos diagramas de seqüência.

Os *diagramas de estado* são criados para documentar os estados de um objeto que tenha um comportamento dinâmico significante. Um diagrama de estado mostra o ciclo de vida de uma classe, os eventos que causam a transição de um estado para o outro, e as ações que resultam de uma mudança de estado.

O *diagrama de atividades* é empregado para a modelagem de aspectos dinâmicos do sistema. Mostra o fluxo de controle de uma atividade para outra.

A seguir veremos todos esses diagramas.

6.4.2.6.1.Diagrama de Seqüência

Conforme dito anteriormente, o diagrama de seqüência dá ênfase à ordenação temporal de mensagens. Os atores se comunicam com os objetos de interface (telas). Os objetos de interface disparam ações com objetos do domínio da aplicação. As ações são convertidas em operações.

Os objetos de interface criados neste estudo de caso, também atuam como objetos de controle (2 camadas) sendo eles responsáveis pela negociação com os objetos de entidade (objetos básicos do domínio).

O Padrão de Projeto simplificado foi utilizado para guiar as escolhas em onde atribuir responsabilidades por ser um modo comum de trabalho para interações simples. A união das classes de interface e controle. As interfaces manipulam as entidades.

Para cada caso de uso criado haverá um diagrama de seqüência que representará os cenários principal e alternativo quando houver.

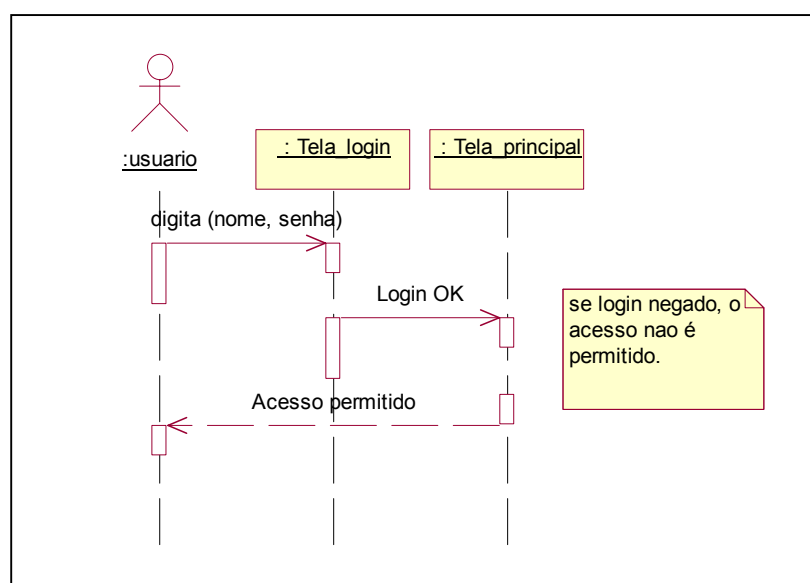



Figura 39 – Diagrama de seqüência: acessar sistema

O diagrama de seqüência acessar sistema, mostrado na Figura 39, inicia com o envio de informações (nome e senha) pelo *usuário*. Somente após essas informações terem sido enviadas, iniciamos a comunicação com os objetos. Ao chamar o método *login* do objeto *Tela principal* este se encarrega de validar as informações passadas e permitir o acesso.

No diagrama de seqüência acessar sistema a nota representa o cenário alternativo. Caso o login seja negado quando as informações forem validadas o acesso não será permitido.

O diagrama de seqüência consultar doador, mostrado na Figura 40, inicia com o envio de informações (nome e data de nascimento) pela *recepcionista*. Somente após essas informações terem sido enviadas, iniciamos a comunicação com os objetos. Ao chamar o método *verifica cadastro* do objeto *Doador* este se encarrega de validar as informações

passadas e listar os dados para o objeto *Tela doador*. Este objeto passará os dados (nome e data de nascimento).Caso o doador não esteja cadastrado haverá uma opção de cadastro. Caso contrário a *recepcionista* selecionará o nome na lista e o objeto *Tela doador* mostrará os outros dados; inclusive a informação se o doador está apto ou não. Caso o doador seja inapto o sistema não permitirá a impressão da ficha. Caso contrário a *recepcionista* pode solicitar a impressão. As notas, no diagrama de seqüência consultar doador, representam o cenário alternativo. Caso o doador não seja cadastrado uma opção de cadastro será apresentada. Caso o doador seja inapto o sistema não permitirá a impressão da ficha para doação.

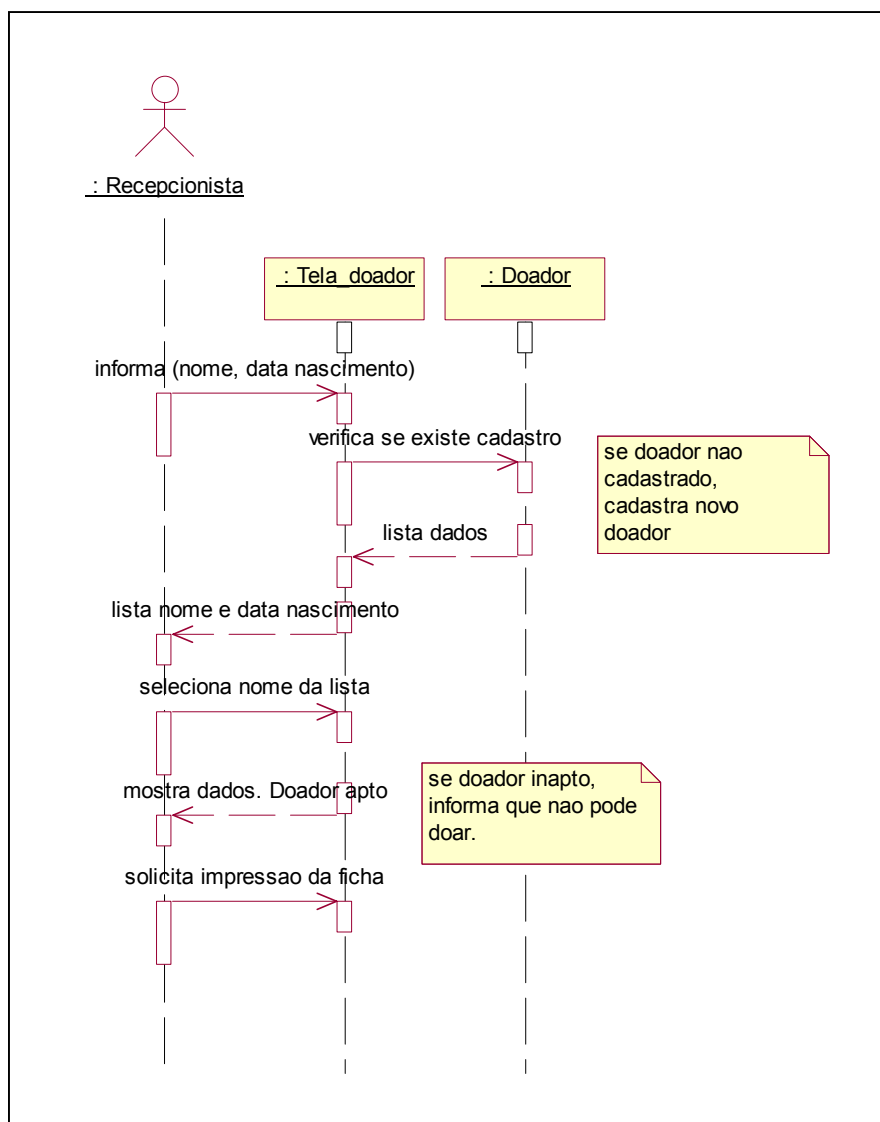



Figura 40- Diagrama de seqüência: consultar doador

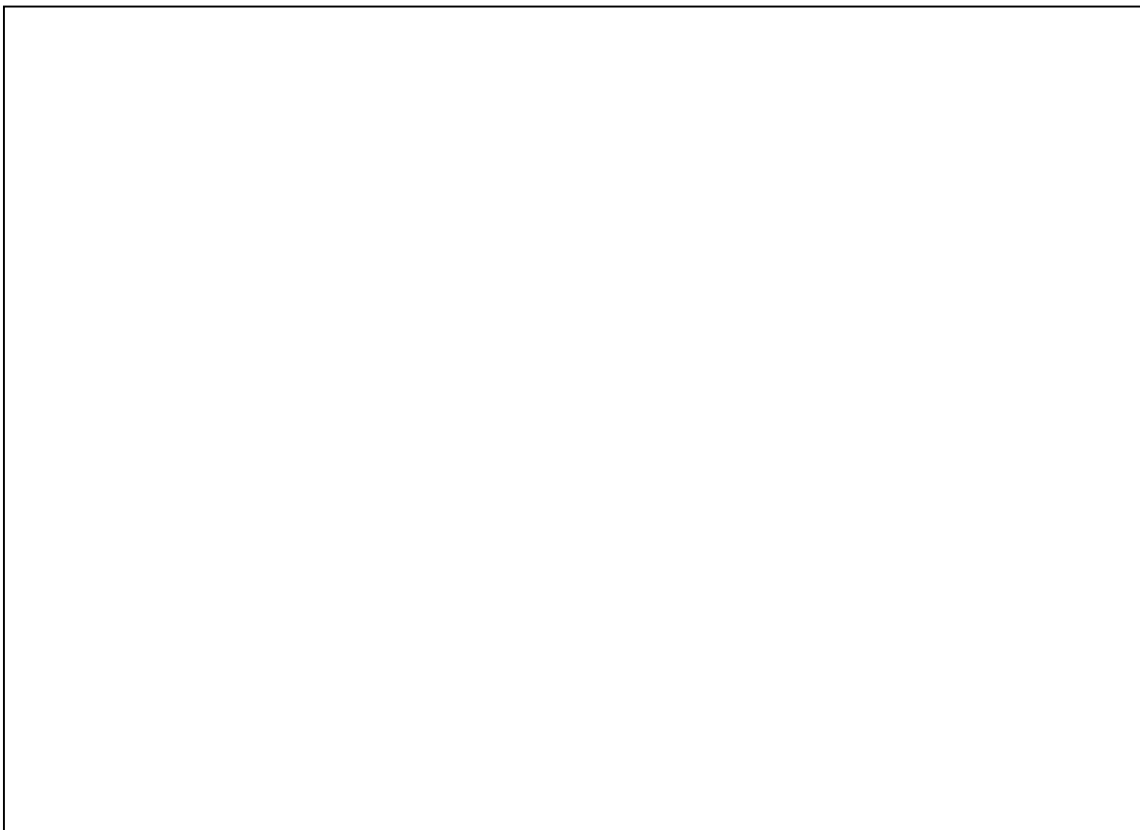



Figura 41 – Diagrama de seqüência: cadastrar novo doador

O diagrama de seqüência cadastrar novo doador, mostrado na Figura 41, inicia com o envio de informações (nome e data de nascimento) pela *recepcionista*. Somente após essas informações terem sido enviadas, iniciamos a comunicação com os objetos. Ao chamar o método *verifica cadastro* do objeto *Doador* este se encarrega de validar as informações passadas e listar os dados para o objeto *Tela doador*. Neste diagrama mostramos o cenário principal quando o doador ainda não tem cadastro.Este objeto passará a mensagem “doador não cadastrado” para a *recepcionista*. A *recepcionista* envia as informações (naturalidade, nacionalidade, estado civil, sexo, rg, cpf, pai, mãe, endereço, cep, cidade, estado, telefone). O método *cadastra novo doador* se encarrega de gravar os dados. Em seguida a recepcionista tem a opção de solicitar a impressão da ficha.

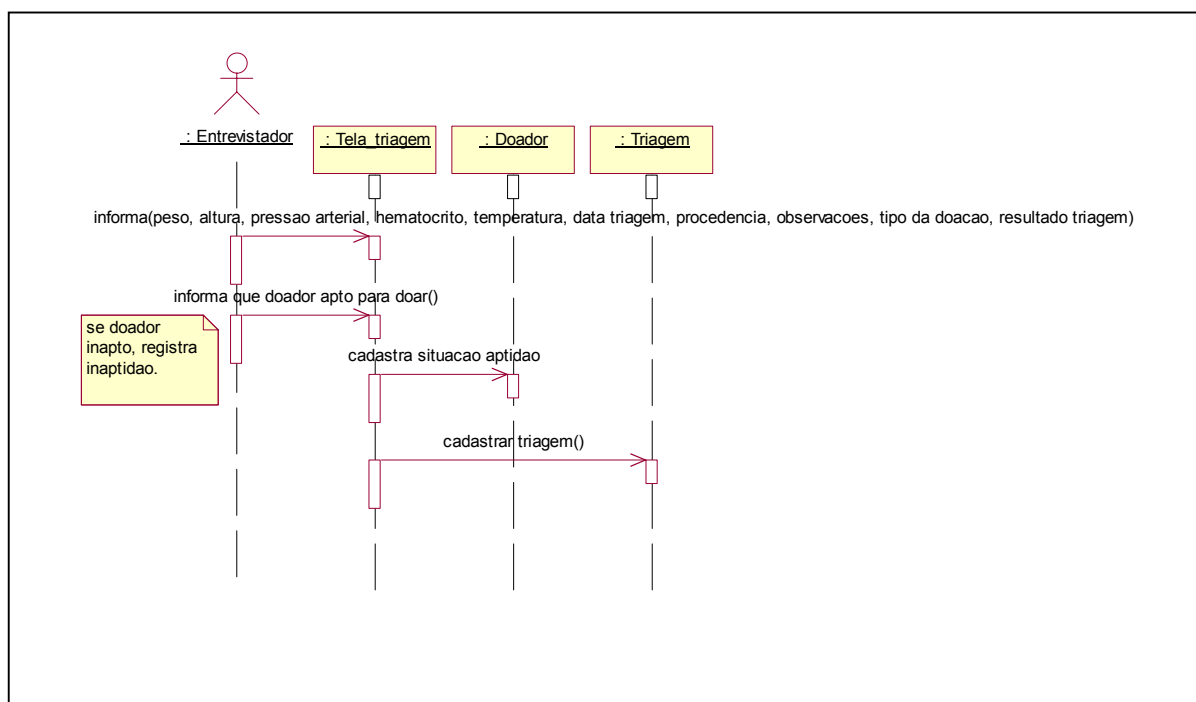



Figura 42 – Diagrama de seqüência: cadastrar triagem

O diagrama de seqüência cadastrar triagem, mostrado na Figura 42, inicia com o envio de informações (peso, altura, pressão arterial, hematócrito, temperatura, data triagem, procedência, observações, tipo da doação, resultado da triagem) pelo *entrevistador*. Somente após essas informações terem sido enviadas, iniciamos a comunicação com os objetos. Ao chamar o método *doador apto a doar* do objeto *doador* este se encarrega de cadastrar a situação de aptidão. Caso o doador seja inapto o objeto *doador* se encarrega de registrar a inaptidão. O objeto *triagem* se encarrega de gravar as informações que lhe são passadas através do método *cadastrar triagem*.

A nota representa o cenário alternativo no diagrama de seqüência cadastrar triagem. Caso o doador seja inapto haverá uma opção para registrar o motivo da inaptidão.

O diagrama de seqüência registrar inaptidão, mostrado na Figura 43, inicia com o envio de informações (peso, altura, pressão arterial, hematocrito, temperatura, data triagem, procedência, observações, tipo da doação) pelo *entrevistador*. Somente após essas informações terem sido enviadas, iniciamos a comunicação com os objetos. O *entrevistador*

se encarrega também de informar a inaptidão. Ao chamar o método *registrar inaptidão* do objeto *triagem* este se encarrega de cadastrar a situação de inaptidão (resultado triagem, motivo de inaptidão). O prazo de inaptidão é validado com o objeto *motivo inaptidão*.A data para nova doação será calculada somando-se o prazo mais a data da triagem. Essa informação será passada para o objeto *doador*. Os outros dados da triagem serão cadastrados pelo objeto *triagem*.

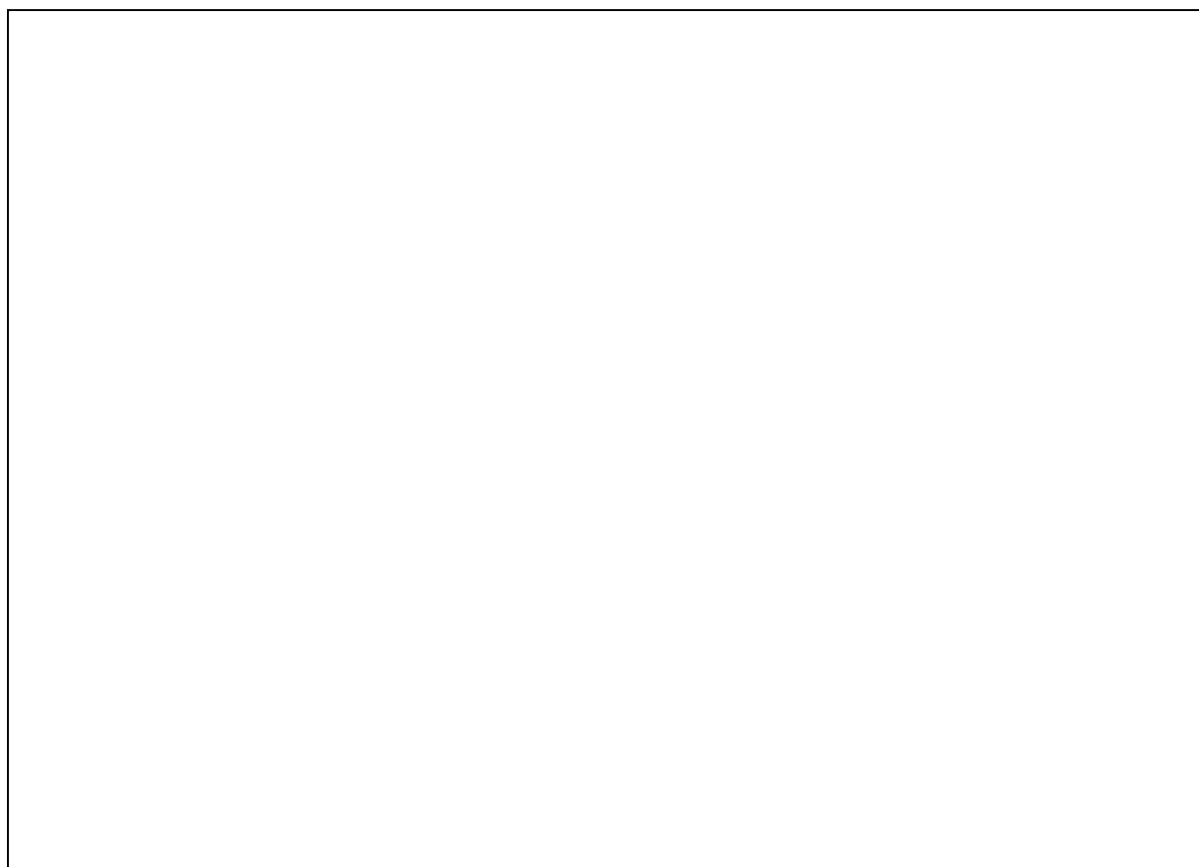



Figura 43 – Diagrama de seqüência: registrar inaptidão

O diagrama de seqüência cadastrar doação, mostrado na Figura 44, inicia com o envio de informações (numero da bolsa, data coleta, hora coleta, volume doador, local de coleta, interior/capital) pelo *responsável pela coleta*. Somente após essas informações terem sido enviadas, iniciamos a comunicação com os objetos. Ao chamar o método *cadastra local doação* do objeto *triagem* este se encarrega de cadastrar (local de coleta, interio/capital). Ao chamar o método *cadastrar doação* do objeto *doação* este se encarrega de cadastrar (numero

bolsa, data coleta, hora coleta, volume doador). Caso o doador tenha alguma reação na hora da coleta será necessário registrá-la. O sistema verificará o número de doações com o objeto *doador* e esta será incrementada de 1.

A nota representa o cenário alternativo no diagrama de seqüência cadastrar doação. Caso o doador tenha alguma reação na hora da coleta de seu sangue, haverá uma opção para se registrar esta reação na hora do cadastro desta doação.

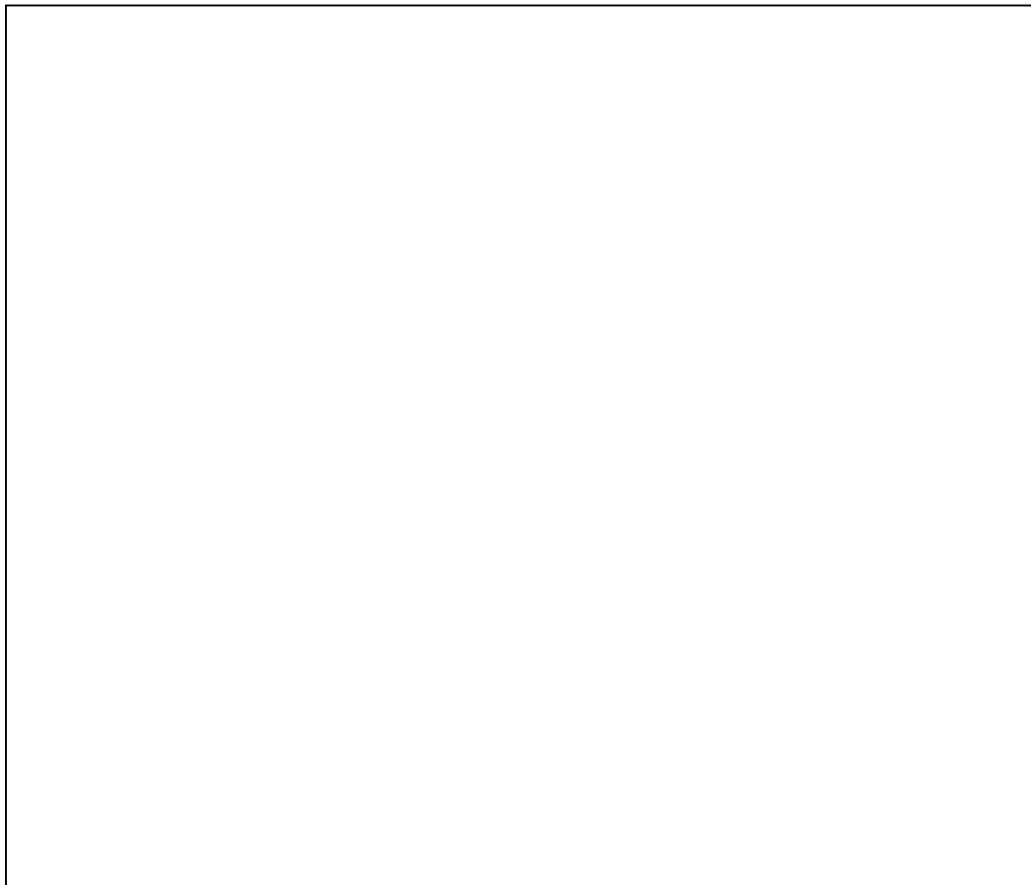



Figura 44 – Diagrama de seqüência: cadastrar doação

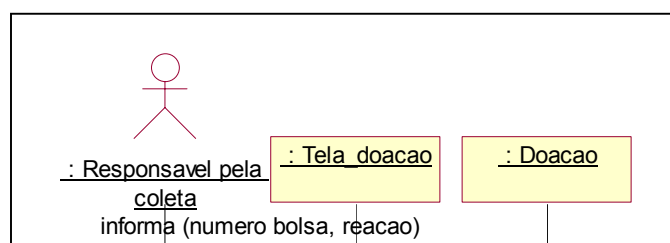



Figura 45 – Diagrama de seqüência: registrar reação

O diagrama de seqüência registrar reação, mostrado na Figura 45, inicia com o envio de informações (numero da bolsa, reação) pelo *responsável pela coleta*. Somente após essas informações terem sido enviadas, iniciamos a comunicação com os objetos. Ao chamar o método *registrar reação* do objeto *doação* este se encarrega de cadastrá-la.

O diagrama de seqüência cadastrar imunohematologia, mostrado na Figura 46, inicia com o envio de informações (numero da bolsa, tipo sanguíneo, fator rh, pesquisa de anticorpos, data imuno, hora imuno, conduta imuno) pelo *técnico de laboratório*. Somente após essas informações terem sido enviadas, iniciamos a comunicação com os objetos. Ao chamar o método *cadastrar tipagem* do objeto *doador* este se encarrega de cadastrá-la juntamente com o fator rh. Ao chamar o método *cadastrar imunohematologia* do objeto *imunohematologia* este se encarrega de cadastrar (pesquisa de anticorpo, data imuno, hora imuno). Ao chamar o método *cadastrar conduta imuno* do objeto *doação* este se encarrega de cadastrar a conduta da imunohematologia.

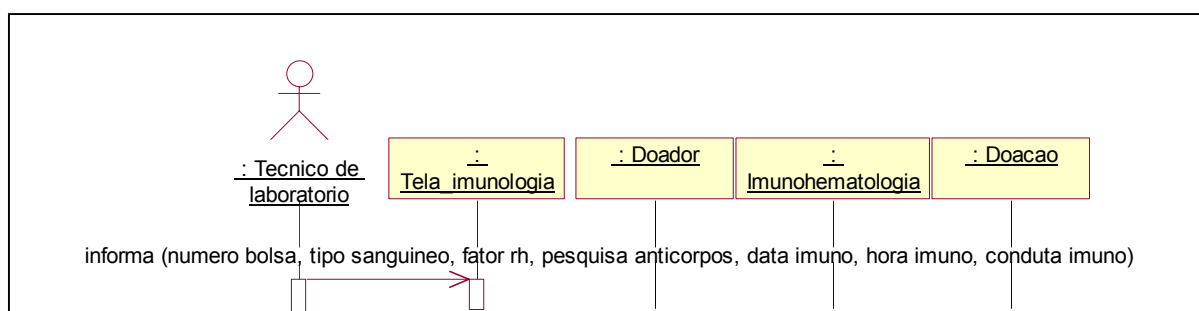



Figura 46 – Diagrama de seqüência: cadastrar imunohematologia

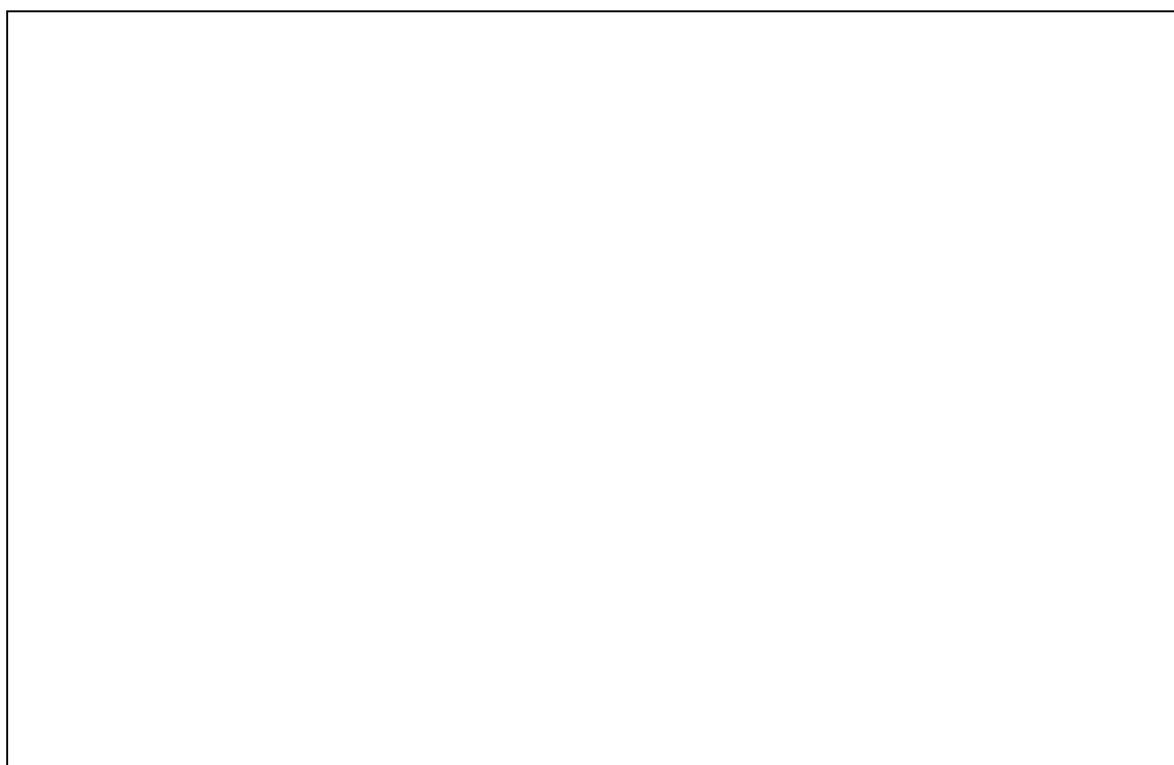



Figura 47 – Diagrama de seqüência: enviar dados para capital

O diagrama de seqüência enviar dados para capital, mostrado na Figura 47, inicia com o envio de informações (período) pelo *técnico de laboratório*. Somente após essas informações terem sido enviadas, iniciamos a comunicação com os objetos. Ao chamar o método *gerar arquivo exportação* do objeto *tela_exportacao_arquivo* este se encarrega de chamar os métodos: *gerar arquivo doador* do objeto *doador*; *gerar arquivo triagem* do objeto *triagem*;

gerar arquivo doação do objeto *doação*; *gerar arquivo imuno* do objeto *imunohematologia*. Cada objeto gerará o arquivo com as informações que estão entre parênteses representadas na Figura. Ao final, quando os arquivos forem gerados o objeto *Tela_exportacao_arquivo* enviara uma mensagem "arquivos gerados" para o ator *técnico de laboratório*.

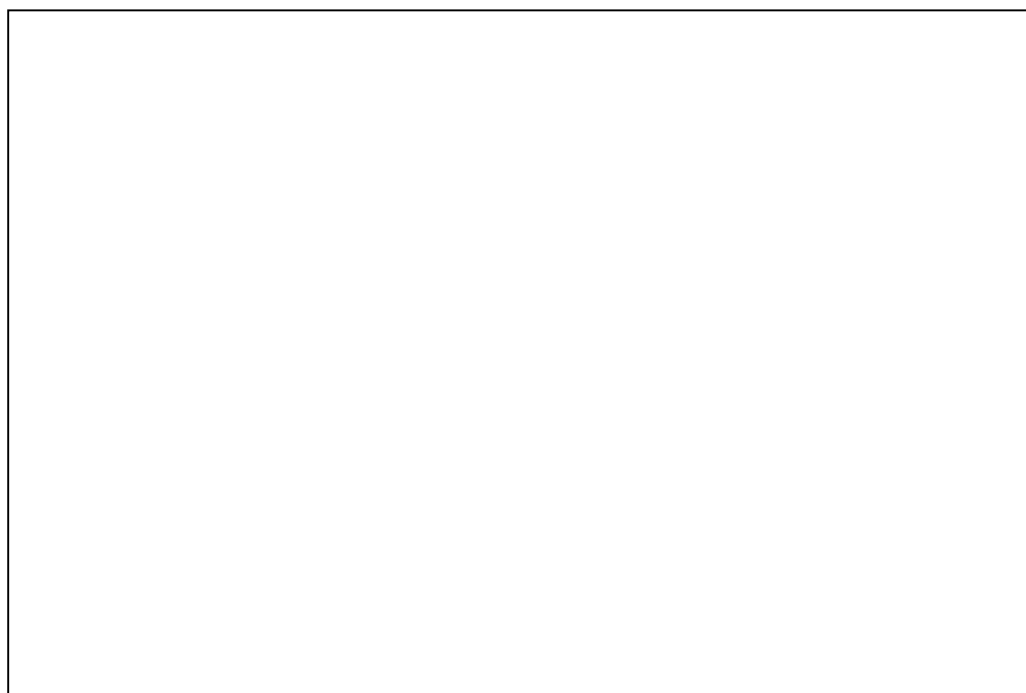



Figura 48 – Diagrama de seqüência: receber dados da capital

O diagrama de seqüência receber dados da capital, mostrado na Figura 48, inicia com o envio de informações (pasta e nome arquivo) pelo *técnico de laboratório*. Somente após essas informações terem sido enviadas, iniciamos a comunicação com os objetos. Ao chamar o método *gerar arquivo importação* do objeto *tela_importacao_arquivo* este se encarrega de chamar os métodos: *importar resultado exame sorologia* do objeto *resultado exame sorologia*; *importar condutas sorologia* do objeto *doação*. O sistema importará os dados e os colocará na base de dados local no interior.

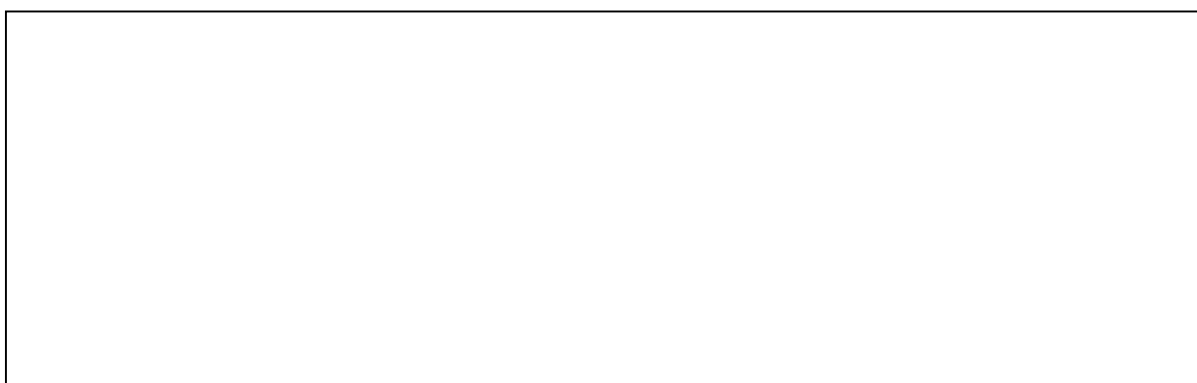



Figura 49 – Diagrama de seqüência: cadastrar geração de produtos

O diagrama de seqüência cadastrar geração de produtos, mostrado na Figura 49, inicia com o envio de informações (numero bolsa, código produto) pelo *técnico do fracionamento*. Somente após essas informações terem sido enviadas, iniciamos a comunicação com os objetos. O objeto *Tela_geracao_produto* se encarrega de chamar os métodos: *verificar data coleta* do objeto *doação*; *verificar validade* do objeto *produto*; *cadastrar geração produto* do objeto *produto gerado*. Se o produto estiver dentro da validade que é o que mostra a Figura, o método *cadastrar geração produto* do objeto *produto gerado* se encarrega de gravar as informações (data geração, hora geração, status do produto). As informações de data e hora da geração são do sistema operacional. A data de vencimento é calculada.

A nota representa o cenário alternativo no diagrama de seqüência cadastrar geração de produtos. Caso na hora da geração do produto ocorrer alguma intercorrência, a mesma será registrada.

O diagrama de seqüência registrar intercorrência, mostrado na Figura 50, inicia com o envio de informações (numero bolsa, código produto, intercorrência) pelo *técnico do fracionamento*. Somente após essas informações terem sido enviadas, iniciamos a comunicação com os objetos. O objeto *Tela_geracao_produto* se encarrega de chamar o

método *registrar intercorrência* do objeto *produto gerado* que se encarrega de gravar as informações informadas e também o status do produto.

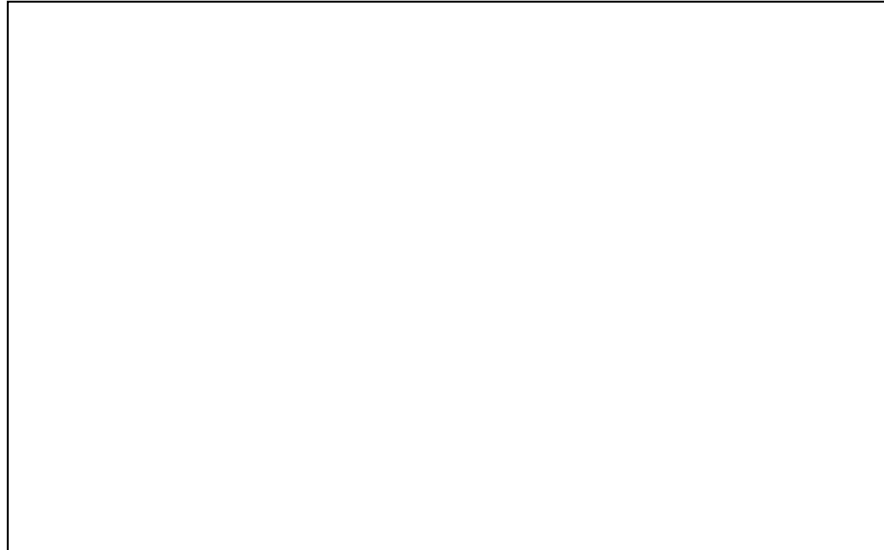



Figura 50 – Diagrama de seqüência: registrar intercorrência

O diagrama de seqüência consultar resultado sorologia, mostrado na Figura 51, inicia com o envio de informações (nome, data de nascimento) pelo *recepcionista*. Somente após essas informações terem sido enviadas, iniciamos a comunicação com os objetos. O objeto *Tela_ doador* se encarrega de chamar o método *verificar resultado exame sorologia* do objeto *resultado exame sorologia* que se encarrega de verificar a existência dos resultados e listá-los se o doador possui o resultado. A *recepcionista* selecionará nome na lista e receberá os resultados. Poderá então imprimi-los.

A nota representa o cenário alternativo no diagrama de seqüência consultar resultado sorologia. Caso o doador ainda não possua resultado de exames, o recepcionista deverá informar ao doador que seus resultados de exames ainda não chegaram da capital.

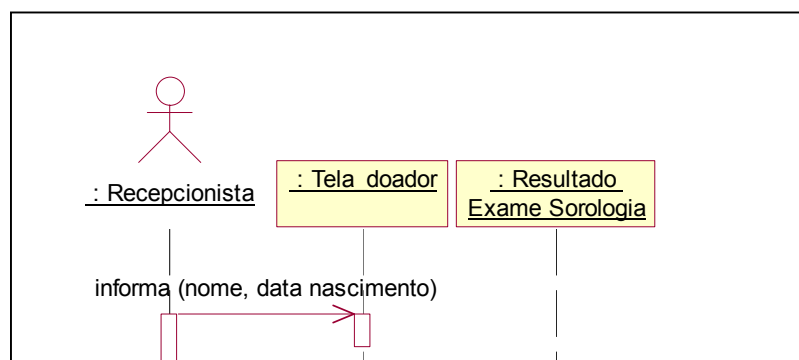



Figura 51 – Diagrama de seqüência: consultar resultado sorologia

O diagrama de seqüência consultar conduta da bolsa, mostrado na Figura 52, inicia com o envio da informação (numero bolsa) pelo *técnico da rotulagem*. Somente após essas informações terem sido enviadas, iniciamos a comunicação com os objetos. O objeto *Tela_ doacao* se encarrega de chamar o método *verificar doacao* do objeto *doacao* que se encarrega de listar a conduta da bolsa seja ela liberada ou desprezada.

As notas representam os cenários alternativos no diagrama de seqüência consultar conduta da bolsa. Quando o método *verificar doação* é chamado se a bolsa não tiver conduta ainda, a mensagem "bolsa sem conduta" será mostrada. O cenário alternativo da conduta liberada é a conduta desprezar bolsa.

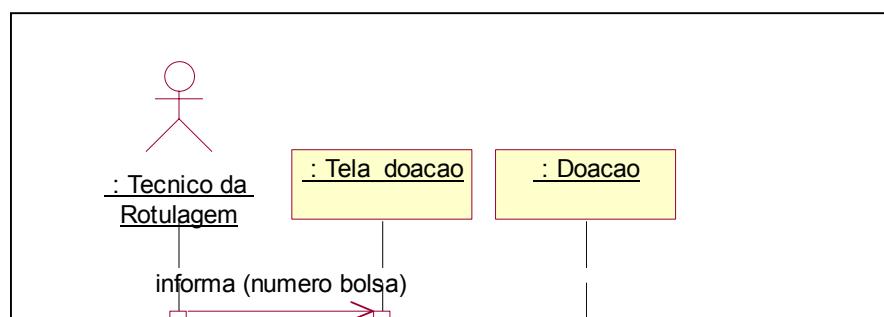



Figura 52 – Diagrama de seqüência: consultar conduta da bolsa

O diagrama de seqüência rotular produto gerado, mostrado na Figura 53, inicia com o envio da informação (numero bolsa, código produto) pelo *técnico da rotulagem*. Somente após essas informações terem sido enviadas, iniciamos a comunicação com os objetos. O objeto *Tela_ rotulagem* se encarrega de chamar o método *verificar validade do produto* do objeto *produto gerado* que se encarrega de informar se a validade está ok ou não. O objeto *Tela_ rotulagem* também se encarrega de chamar o método *verificar conduta da bolsa* do objeto *doacao* que se encarrega de informar se a conduta está liberada ou não. Se tudo estiver ok, o objeto *Tela_rotulagem* diponibilizará a opção de impressão da etiqueta. A etiqueta será impressa. O objeto *Tela_rotulagem* chamará o método *cadastrar status do produto* do objeto *produto gerado* que gravará os dados: status do produto; data rotulagem; hora rotulagem. Essas datas serão do sistema operacional.

As notas representam os cenários alternativos no diagrama de seqüência rotular produto gerado. Caso a validade estiver não ok a etiqueta não será impressa e o status do produto receberá “descarte por vencimento”. Caso a conduta informada seja “desprezada”, quando o

método consulta conduta da bolsa é chamado, a etiqueta não será impressa e o status do produto receberá “descarte por exame”.

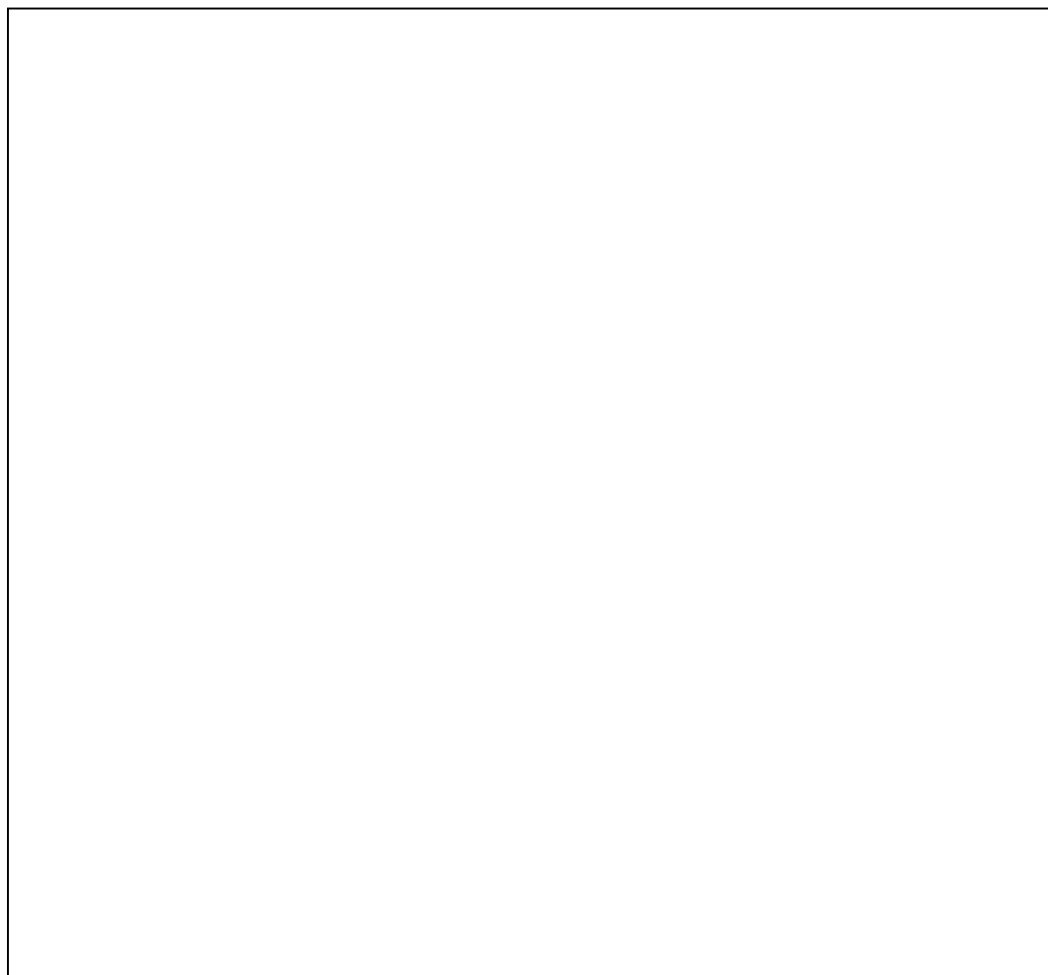



Figura 53 – Diagrama de seqüência: rotular produto gerado

O diagrama de seqüência cadastrar distribuição do produto, mostrado na Figura 54, inicia com o envio da informação (numero bolsa, código produto) pelo *técnico da distribuição*. Somente após essas informações terem sido enviadas, iniciamos a comunicação com os objetos. O objeto *Tela_ distribuição* se encarrega de chamar o método *verificar doacao* do objeto *doacao* e *verificar produto* do objeto *produto gerado* que se encarregam de verificar se a doação e produto estão ok. Se ambos estiverem ok, o objeto *Tela_distribuicao* chamará o método *cadastrar distribuição produto* do objeto *produto gerado* que se

encarregará de gravar os dados: data distribuição, hora distribuição, status do produto=distribuído. Caso contrário a mensagem “distribuição não permitida” será mostrada.

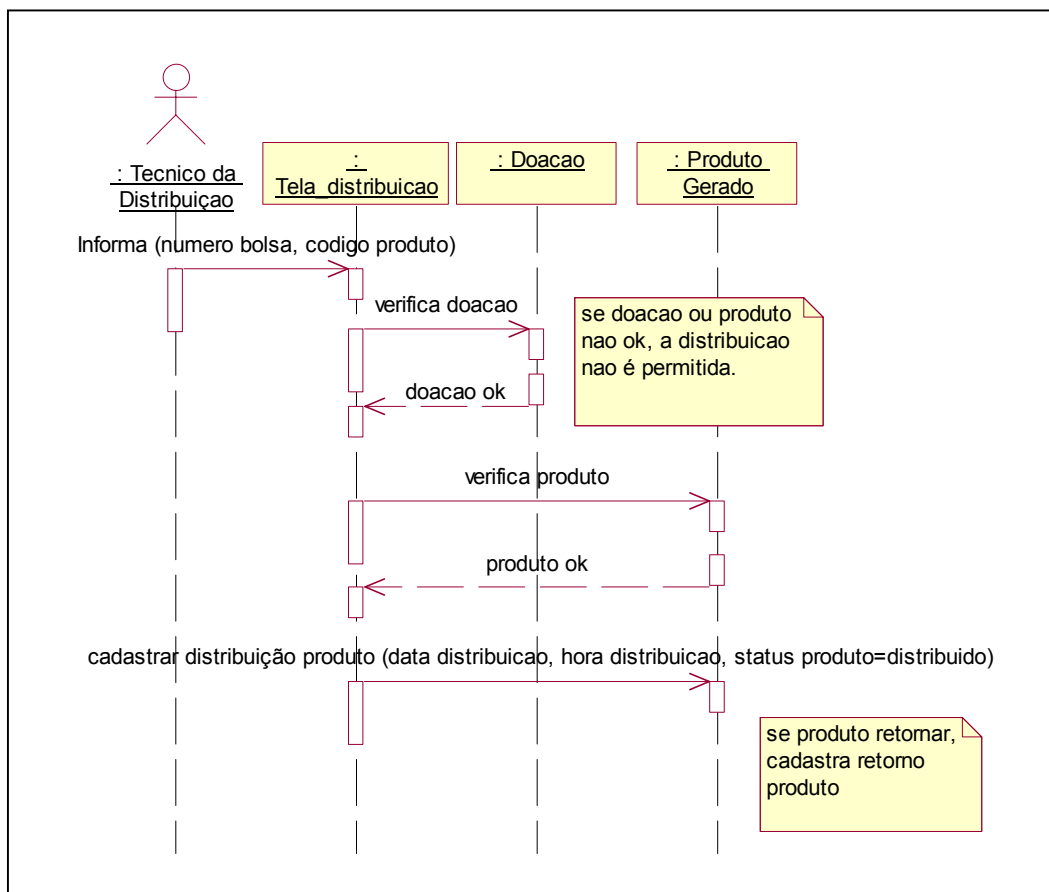



Figura 54 – Diagrama de seqüência: cadastrar distribuição do produto

As notas representam os cenários alternativos no diagrama de seqüência cadastrar distribuição do produto. Caso a doação ou o produto não estiverem ok, sua distribuição não será permitida. Quando o produto for distribuído e retornar haverá o cadastro do retorno deste produto.

O diagrama de seqüência cadastrar retorno do produto, mostrado na Figura 55, inicia com o envio da informação (numero bolsa, código produto) pelo *técnico da distribuição*. Somente após essas informações terem sido enviadas, iniciamos a comunicação com os objetos. O objeto *Tela_ retorno* se encarrega de chamar o método *cadastrar devolucao do*

produto do objeto *produto gerado* que se encarrega de gravar as informações: data retorno, hora retorno, status do produto=retornado.

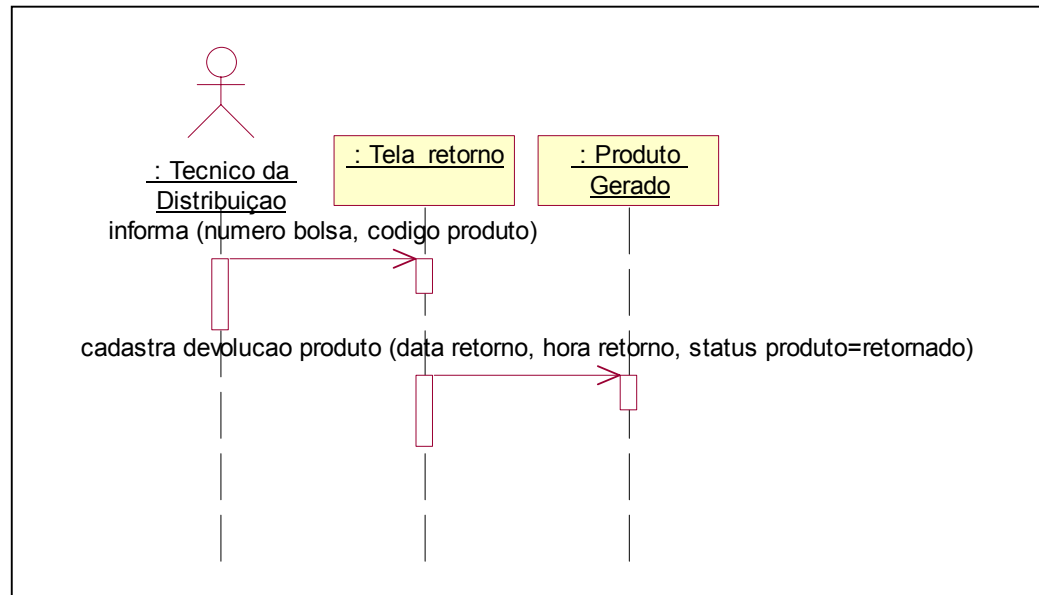



Figura 55 – Diagrama de seqüência: cadastrar retorno do produto

O diagrama de seqüência cadastrar descarte por validade, mostrado na Figura 56, inicia com o envio da informação (numero bolsa, código produto) pelo *técnico da distribuição*. Somente após essas informações terem sido enviadas, iniciamos a comunicação com os objetos. O objeto *Tela_ descarte_validade* se encarrega de chamar o método *cadastrar descarte produto* do objeto *produto gerado* que se encarrega de gravar as informações: data descarte, hora descarte, status do produto=descartado por vencimento.

Após a criação dos diagramas de seqüência, voltaremos ao diagrama de classes para incluir as operações. Na próxima seção mostraremos o diagrama de classes com as operações.

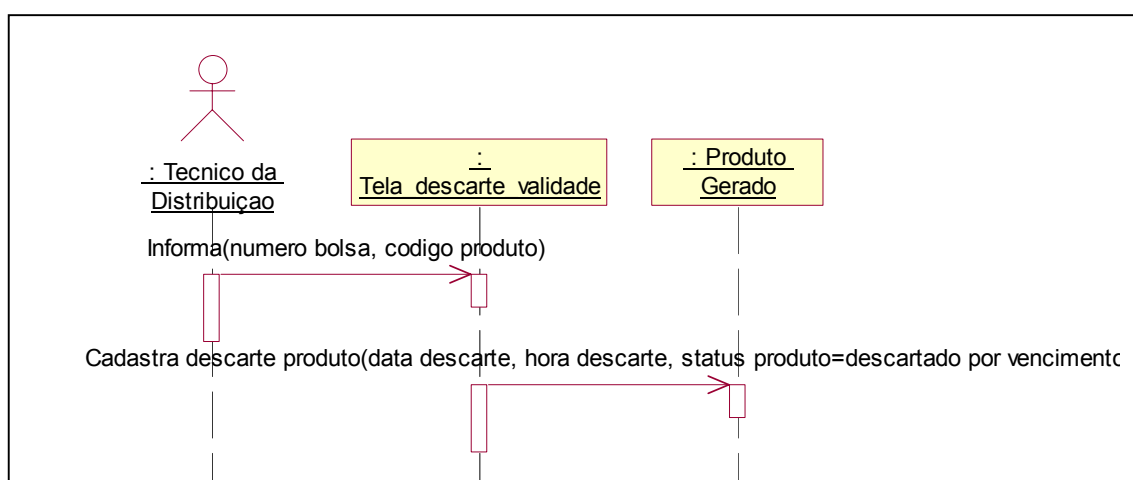



Figura 56 – Diagrama de seqüência: cadastrar descarte por validade

6.4.2.6.2.Diagrama de Classes de Projeto

Na fase de projeto o diagrama de classes é completado com as operações descobertas através dos diagramas de seqüência. Ver Figura 57.

A Figura 57 mostra o diagrama de classes e suas operações. Ressaltamos que, nem todas as operações encontradas na elaboração dos diagramas de seqüência foram colocadas nesta figura, colocamos apenas algumas operações.

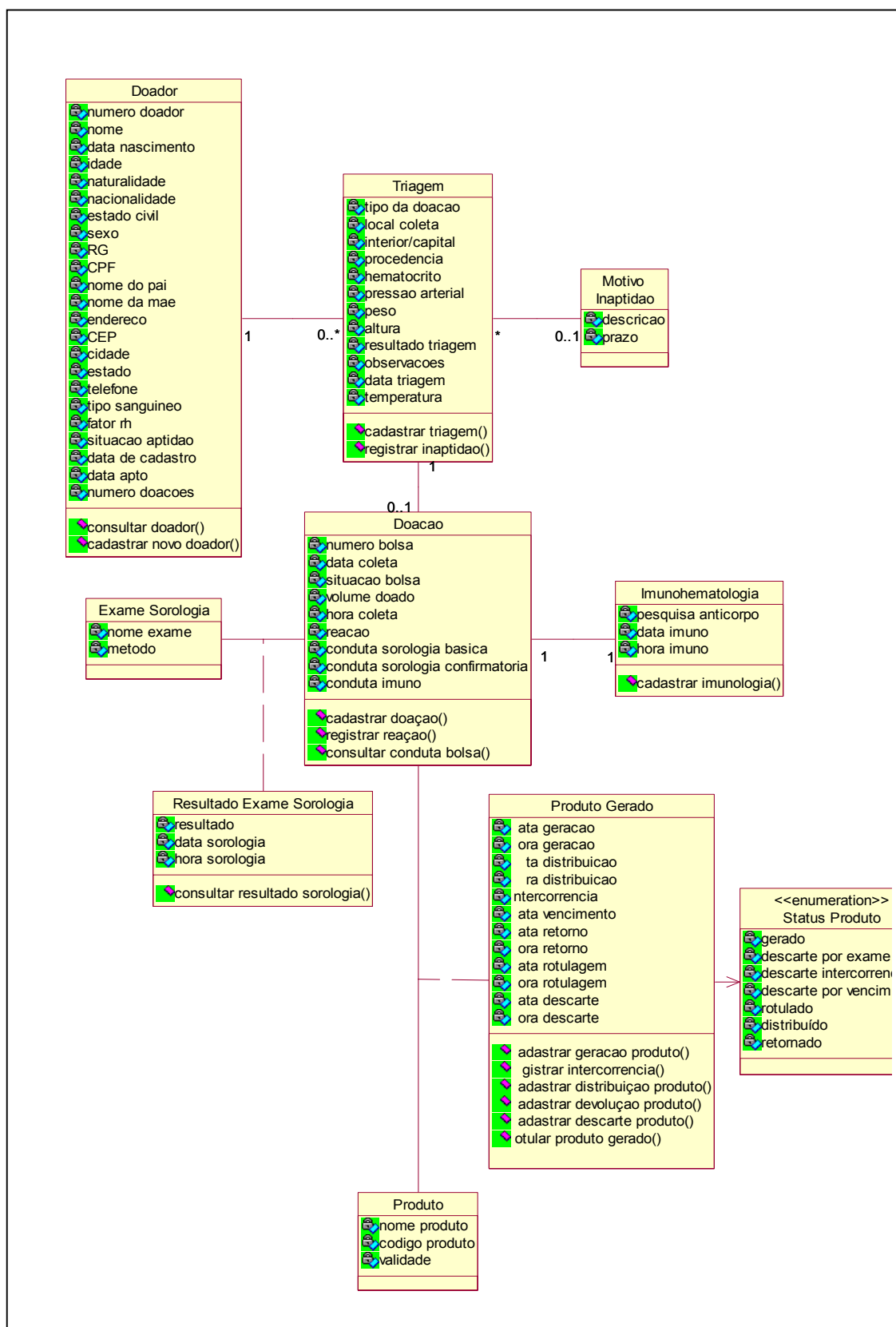



Figura 57 – Diagrama de classes e suas operações

6.4.2.6.3.Diagrama de Estados

Existem dois objetos que possuem comportamento interessante, o objeto doador e o objeto produto gerado. Para esses objetos projetamos o diagrama de estado. Antes de explicarmos os diagramas de estado criados, é necessária uma breve explicação sobre esses objetos.

O doador doa seu sangue. O sangue é coletado em uma bolsa. Esta bolsa é fracionada e o sangue nela contido é separado gerando produtos.

O diagrama de estado da classe doador, Figura 58, representa a situação do doador. Inicialmente, após o cadastro na recepção o doador torna-se *apto a ser triado*, se ele for aprovado estará *apto a doar sangue*, caso contrário receberá estado de *inapto temporário*, então poderá passar por consulta médica ou não, dependendo do seu motivo de inaptidão. Quando o doador está *apto a doar*, ele faz a doação e seu estado muda para *aguardando resultado* até que seus exames laboratoriais estejam concluídos. O doador recebe o resultado de seus exames e caso esteja tudo normal, passa para o estado de *inapto temporário* até que seu prazo de inaptidão se encerre e ele possa doar novamente (3 meses para mulheres e 2 meses para homens). Se der alguma anormalidade no resultado de seus exames, o doador passa a ter o estado de *com problemas na doação anterior* e ele é encaminhado para uma consulta médica. Dependendo de seu problema, o médico pode liberar ou não o doador para novas doações *apto a ser triado*, deixá-lo *inapto temporario* ou *inapto definitivo*. O estado de *inapto definitivo* impossibilita o doador de doar sangue novamente. O atributo *situação aptidão* da classe *doador* guarda os estados do doador. Quando o doador é inapto, por qualquer motivo, este motivo será guardado na classe *motivo de inaptidão*.

O diagrama de estado, mostrado na Figura 59, representa a situação da bolsa de sangue colhida na doação e seus vários produtos, até o momento em que, esses produtos gerados a partir da bolsa, saem do Hemoam.

Inicialmente, a bolsa de sangue, após a coleta, fica com o estado de *bolsa coletada*. Em seguida, a bolsa é encaminhada ao setor de Fracionamento onde é fracionada em alguns produtos. Cada produto gerado a partir da bolsa fica com estado de *produto gerado*. Se no ato de fracionar a bolsa de sangue acontecer alguma intercorrência aquele produto que está sendo gerado é desprezado e fica com o estado de *produto desprezado*. Os exames laboratoriais são realizados (imuno e sorologia) e lançadas condutas para bolsa. Quando a conduta da bolsa é liberada seus produtos são liberados para Rotulagem seu estado é *produto liberado*. Se a conduta for desprezar o produto é desprezado e fica com o estado de *produto desprezado*. Após a rotulagem o produto é estocado e fica com o estado de *produto estocado*. Uma vez estocado, o produto pode ser *distribuído* ou *desprezado* se seu prazo de validade vencer. Quando o produto é distribuído, existe a possibilidade dele retornar, se isso acontecer ele ficara no estado de *produto sendo verificado* e após a verificação ele poderá ser estocado novamente ou ser desprezado.

Os estados do objeto doador ocorrem concorrentemente aos estados do objeto produto gerado. Quando o resultado dos exames do doador ficam prontos, o produto gerado muda do estado de gerado, para liberado (resultado sem problema) ou desprezado (problema no resultado), dependendo do resultado do exame do doador. Por serem objetos distintos os diagramas de estados foram criados separados. Eles poderiam ser melhor observados através do diagrama de atividades que falaremos mais adiante.

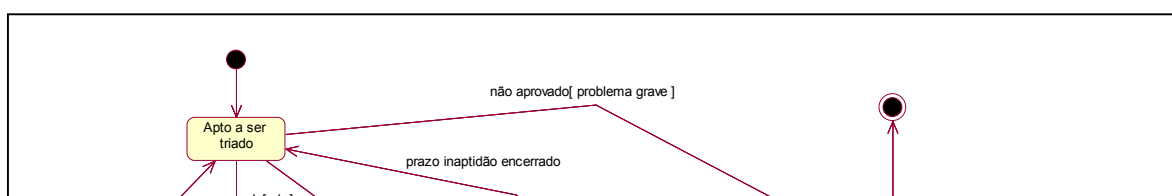



Figura 58 – Diagrama de estado: situação doador

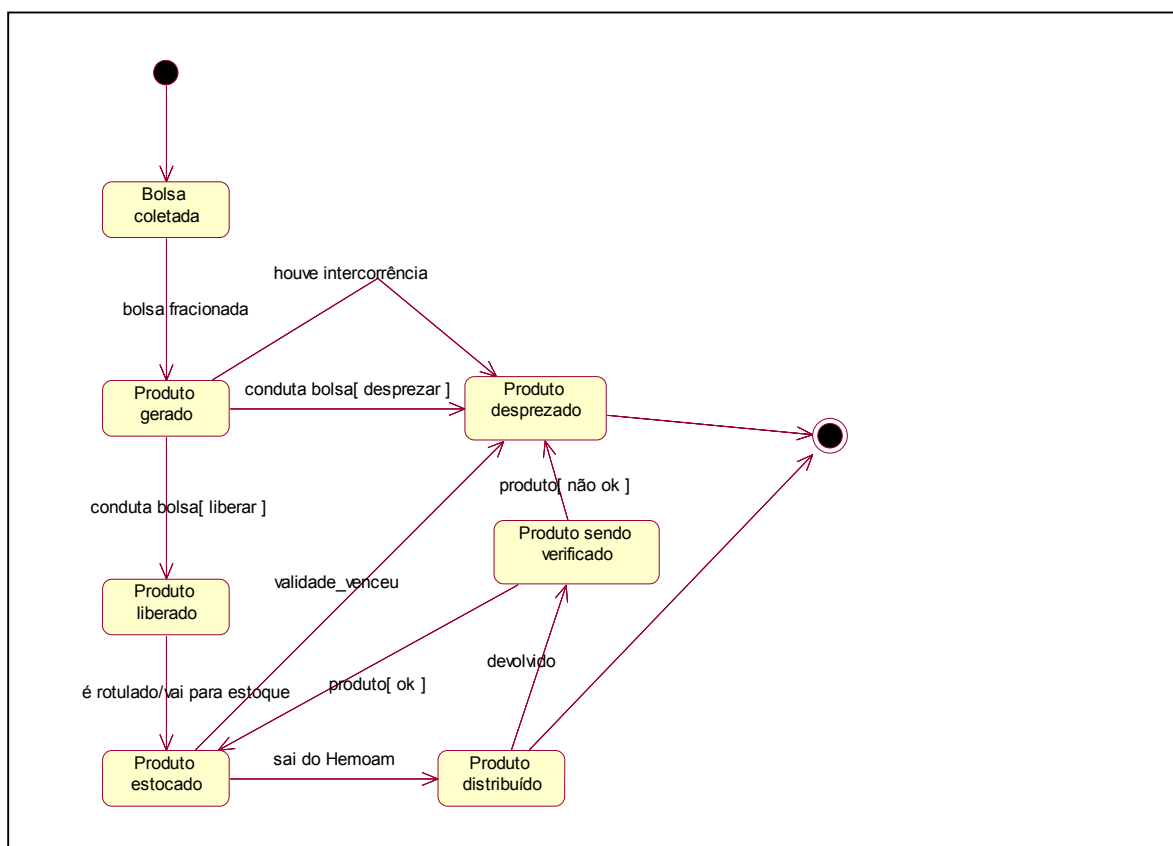



Figura 59 – Diagrama de estado: situação da bolsa e seus produtos

6.4.2.6.4.Diagrama de Atividades

No diagrama de atividades, utilizado para fazer a modelagem do fluxo de trabalho, foram utilizadas raias (swinlanes). Cada raia representa um grupo da organização responsável por estas atividades. Os grupos são: recepção de doadores, hematócrito, triagem, coleta, fracionamento, imunologia, sorologia na capital, rotulagem e distribuição.

O diagrama de atividades, Figura 60 (em anexo), foi elaborado com base na descrição do estudo de caso seção 6.4.2.1.

A seguir vamos finalizar a modelagem do sistema SADI com a modelagem arquitetural. Nela mostraremos dois diagramas: *diagrama de componente* e *diagrama de implantação*.

6.4.2.7.Modelagem Arquitetural do Sistema SADI

O objetivo da modelagem arquitetural é descrever a arquitetura física do sistema. Seu foco é o mapeamento da estrutura lógica de classes para uma arquitetura física em termos de componentes e de nós de processamento.

A arquitetura física descreve a decomposição detalhada do hardware e software que compõem a implementação do sistema. Os diagramas utilizados são os *diagramas de implementação* onde são mostrados os dispositivos de hardware envolvidos, as questões de conexão de redes, a localização física das classes e objetos, a localização física do banco de dados, as dependências de arquivos, etc. Existem dois tipos de diagramas de implementação: *diagrama de componentes* e *diagrama de implantação*. O diagrama de componente mostra a estrutura do próprio código fonte. O diagrama de implantação mostra a estrutura do sistema *run-time*.

6.4.2.7.1.Diagramas de Implementação

Conforme o que foi apresentado na plataforma do sistema SADI haverá duas estruturas, a estrutura na UCT do interior e a estrutura da capital. Os diagramas de implementação mostram os aspectos de implementação física, incluindo a estrutura de componentes. Eles são expressos de duas formas: diagrama de componentes e diagrama de implantação.

6.4.2.7.2.Diagrama de Componente

Conforme mostra a Figura 61, o componente principal *SAD-Interior.exe* contém rotinas básicas de tratamento dos dados, como acesso ao banco local. As rotinas mais específicas estarão dispostas no componente **Interior.dll*, i.e. validação das classes persistentes, o fluxo de negócio entre elas, rotinas de transmissão do arquivo txt. O mesmo é válido para os componentes dispostos na capital.

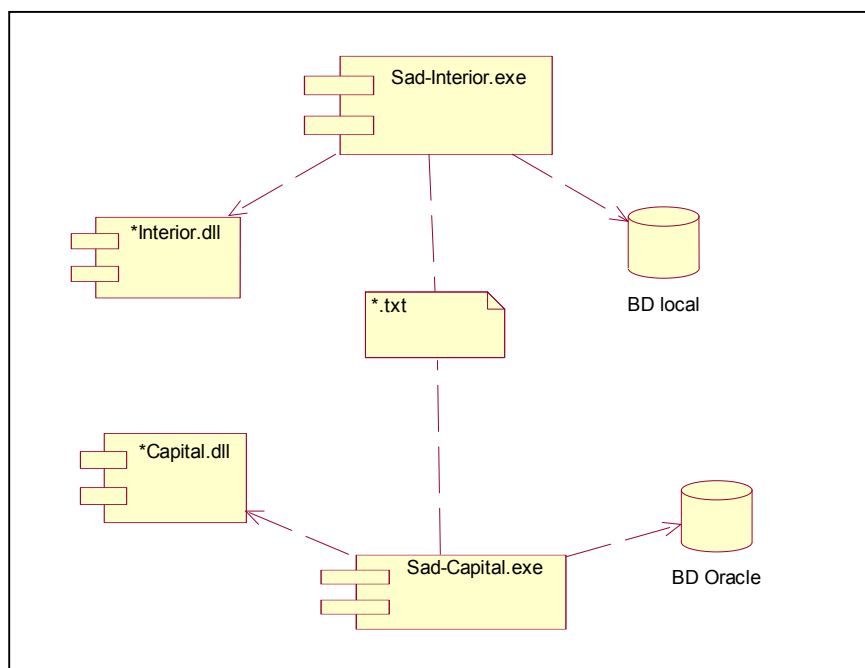



Figura 61 – Diagrama de componente

6.4.2.7.3.Diagrama de Implantação

A Figura 62 mostra, o nó *Estação da UCT* que conterá os componentes SAD-Interior e as bibliotecas dlls referentes a ele, o nó *Servidor* que contêm componentes que recebem/enviam dados através do componente SAD-Capital e suas respectivas dlls que alimentam a base de dados representada pelo nó *Servidor de BD*.

O nó *Estação da UCT* se relacionará com o nó *Servidor*, através da Internet, para transmissão de dados.

Figura 62 – Diagrama de implantação

Entre o nó *Servidor* e o nó *Servidor de BD* a ligação é feita via rede local padrãp 100 BASE T Ethernet.

CONSIDERAÇÕES

Neste capítulo, estudo de caso, passamos por duas etapas: *análise* e *projeto*. Em ambas etapas pudemos ver diversos modelos, cada qual com seus diagramas específicos que nos permitiram modelar diversos elementos do sistema SADI. O que está faltando agora é partirmos para as próximas etapas: *implementação*, *verificação* e *validação*, *implantação* e *manutenção*. Mas essas não fazem parte do escopo deste trabalho, elas ficarão para trabalhos futuros.

CAPÍTULO 7 – CONCLUSÕES E TRABALHOS FUTUROS

O desenvolvimento desse trabalho foi realizado com ênfase na modelagem do sistema SADI com apoio da UML.

A ênfase na modelagem do sistema SADI foi estabelecida na metade do curso (12 meses) quando iniciamos a pesquisa. Desde esse tempo, nosso foco foi a modelagem de um sistema que atendesse às necessidades vislumbradas de início na empresa.

A escolha da UML como linguagem de modelagem deu-se pelo fato de ser independente de linguagem de programação e processo de desenvolvimento, o que nos proporcionou liberdade na escolha do ciclo de vida de desenvolvimento do nosso software, e também, por ser uma linguagem bastante conhecida tanto no meio profissional quanto acadêmico.

A escolha da orientação a objeto foi por ser *a melhor abordagem para criar e manter software de maneira rápida e barata*, segundo os autores que pesquisamos. Eles afirmam que isso se dará graças ao emprego de poderosos conceitos e abstrações e também pela perspectiva de promover reusabilidade em grande escala dos componentes modelados. Nossa grande preocupação, na escolha deste paradigma, foi a de garantir compatibilidade entre as fases de análise e projeto e posteriormente do projeto para implementação.

A preocupação com o processo de software surgiu com o passar do tempo, sentimos necessidade de estabelecer um processo iterativo de desenvolvimento para a organização. Nossa necessidade era entender, avaliar, controlar, aprender, comunicar, melhorar, predizer e certificar o nosso trabalho como desenvolvedores de software. Para isso foi preciso analisar, avaliar, comparar os processos existentes para então definir o processo que iríamos adotar,

visto que este tem se mostrado o fator determinante para o alcance da qualidade do produto final.

Inicialmente a idéia era adotar o Rational Unified Process (seção 4.1.1.4.1) no desenvolvimento do software de nosso estudo de caso, mas após estudá-lo, percebemos que nos facilitaria ter um processo mais simples, com etapas bem definidas, e que nos possibilitasse gerenciar estas atividades. Diante disso, resolvemos apenas nos basear no Rational Unified Process criando um Processo Próprio (seção 6.4.2). Das etapas definidas no processo (análise, projeto, implementação, verificação e validação, implantação e manutenção) o trabalho cobriria as 2 etapas iniciais, em função dos prazos, deixando-se para trabalhos futuros a implementação das demais etapas e a revisão do próprio processo de desenvolvimento definido.

Uma limitação do trabalho é, até o momento, que não houve reuso das classes modeladas, pois como dissemos, o trabalho ainda encontra-se nas etapas iniciais do processo (análise e projeto). É uma expectativa nossa, mas não sabemos se irá ocorrer realmente.

Com a utilização do processo definido de desenvolvimento no HEMOAM, nas etapas iniciais, aumentamos nossa experiência em desenvolvimento; inserimos modificações no processo de desenvolvimento utilizado na Fundação Hemoam; o processo passou a ser conhecido por todos da equipe de informática; papéis e responsabilidades ficaram bem claros; melhoramos a facilidade de mudanças de requisitos que é bem difícil de se conseguir; demos ênfase na criação e manutenção de modelos para que as informações pudessem ser visualizadas, especificadas e capturadas instantaneamente e controladas eletronicamente melhorando a comunicação entre a equipe de desenvolvimento e a gerência de sistemas; e aumentamos a participação dos usuários do sistema no processo de desenvolvimento.

A modelagem dentro do processo de desenvolvimento nos permitiu entender a informação, a função, o comportamento e a completude do sistema, tornando a tarefa de

análise de requisitos mais completa. Os requisitos foram todos documentados o que não acontecia antes da modificação do processo de desenvolvimento. Podemos dizer que a modelagem serviu de base para o projeto fornecendo uma representação essencial do software que no futuro será mapeada para o contexto de implementação. Vale ressaltar, neste ponto que, a busca de boas representações, capazes de realizar mapeamentos precisos entre os níveis conceitual, lógico e físico, com a eficiência, economia e expressividade desejadas, representa um permanente desafio aos pesquisadores da área.

A modelagem nos permitiu, ainda, documentar as decisões que foram tomadas ao longo das fases de análise e projeto do sistema, manter estas decisões atualizadas e também de introduzir modificações nessa documentação. Entendemos que dando ênfase na modelagem aumentamos a probabilidade de diminuir erros em etapas futuras no processo de desenvolvimento, pois na maioria das vezes, os erros são provenientes de má especificação, de falta de entendimento sobre os objetivos a serem alcançados, e de resultados de traduções imperfeitas de requisitos e especificações que provocam a maior parte dos problemas.

O uso dos diagramas da UML facilitou a comunicação com nossos usuários internos, pois os diagramas são bem fáceis de entender e com isso temos conseguido validar nossos requisitos de sistema de forma mais precisa. Outro benefício, foi poder visualizar o sistema em níveis diferentes de detalhamento em situações distintas, por exemplo, a visão dos casos de uso (expondo os requisitos do sistema), a visão de projeto (captando o vocabulário do espaço do problema e do espaço da solução), a visão do processo (ordenando as atividades e definindo papéis e responsabilidades para um melhor controle do desenvolvimento do sistema).

O término deste trabalho foi de suma importância na aquisição de novos conhecimentos sobre metodologias e tecnologias para o desenvolvimento de software, vale ressaltar que foi

gratificante aplicar o conhecimento adquirido no estudo de caso, contribuindo desta forma com a instituição e a própria comunidade do interior do estado.

Quanto à possibilidade de trabalhos futuros é interessante destacar quatro pontos. O primeiro diz respeito às etapas de implementação, verificação e validação, implantação, e manutenção do sistema SADI, que possibilitará a troca de informações com o sistema da capital, havendo com isso aumento da agilidade e segurança no processo da doação de sangue no interior do estado do Amazonas. O segundo ponto seria implantar o sistema em outros hemocentros que ainda não estão informatizados, contribuindo assim, com outras instituições. O terceiro ponto, é no futuro, incrementarmos o processo de desenvolvimento criado, quando as outras etapas do processo forem realizadas, com o objetivo de melhorar cada vez mais a qualidade dos sistemas desenvolvidos na Fundação HEMOAM. E finalmente, o último ponto seria disponibilizar este sistema on-line (web) quando a quantidade de informações trocadas entre o interior e a capital aumentarem.

BIBLIOGRAFIA

BASILI, V. e Zelkowitz, M.. **Analysing Médium Scale Software Development, Proc**. 3rd Intl. Conf. Software Engineering, IEEE, 1978, pp. 116-123.

BASS, L.; CLEMENTS, P. ; KAZMAN, R. **Software architecture in pratice**. Massachusetts, Addison-Wesley Publishing Company, 1998.

BLANCHARD, B. S. ; FABRYCKY,W. J. **Systems Engineering and Analysis**. Prentice-Hall, 1981.

BOEHM, B. W. **Software Risk Management**, IEEE Computer Society Press, 1989.

BROWN, K. ; WHITENACK, B. 1996. **Crossing Chasms. Pattern Languages of Program Design**. vol. 2. Reading, MA.: Addison-Wesley.

CHARETTE, R. N. **Software Engineering Risk Analysis and Management**. McGraw-Hill/Intertext, 1989.

COUGO, Paulo. **Modelagem conceitual e projeto de banco de dados**. Rio de Janeiro: Editora Campus ltda, 1997.

DEVELOPMENT Process. Disponível em: <http://seed.edrc.cmu.edu/SL/SL-devlproc.html>. Acesso em 23 de abril de 2003.

Documentações sobre a UML. Disponível em: <http://www.omg.org>. Acesso em jan. 2002.

Documentações sobre a UML. Disponível em: <http://www.uml.org>. Acesso em jan. 2002.

DOWNS, C.; COE.**SSADM, Aplication and Context**. 2rd. Edition. Prentice-Hall, Inc., 1991.

CURTIS, B. **The global pursuit of process maturity**. IEEE Software, jul./ago. 2000.

HUMPHREY, W. S. **Introduction to the Team Software Process**. Addison-Wesley, Reading – MA, 1999.

KING, J.; SCHREMS, E. **Cost Benefit Analysis in Information Systems Development and Operation**. ACM Computing Surveys, vol. 10, no. 1, março de 1978, pp. 19-34.

LINDVALL, M.; RUS, I. **Process diversity in software development**. IEEE Software, v.17, n.4, jul./ago. 2000.

MALCON, E. **SSADM, Version 4: a user's guide**. McGraw-Hill, 1994.

MCDERMID, J.A.; ROOK, P. **Software development process models**. Software Engineering Reference Book, ed. J.A. McDermid, Butterworth-Heineman Scientific, 1991.

PÁDUA, W. **Engenharia de Software – fundamentos, métodos e padrões**. Livros Técnicos e Científicos Editora. Rio de Janeiro, 2001.

PFLEEGER, S.L. **Software engineering – theory and pratice**. Nova Jersey: Prentice-Hall Inc. 1998.

PRESSMAN, R. S. **Software engineering – a practitioner´s approach**, 5a. edição. Nova York: McGraw-Hill, 2000.

RAMIREZ, A.; VANPEPERSTRAETE, P.; RUECKERT, A.; ODUTOLA, K.; BENNETT, J. **ArgoUML User Manual**. Disponível em: <http:// argouml.tigris.org />. Acesso em 17 de junho de 2003.

SHAW, M. ; GARLAN, D. **Software architecture: perspectives on an emerging discipline**. Nova Jersey: Prentice-Hall, Inc., 1996.

SOMMERVILLE, I. **Software engineering (international computer science series)**. Massachusetts: Addison-Wesley Publishing Company, ago. 2000.

SPIVEY, J.M. **The Z notation: a reference manual**. Nova Jersey: Prentice-Hall Inc., 1989.

SYSTEM Architect V9.; WHITEPAPERS. Disponíveis em: <http://www.popkin.com>. Acesso em: 17 de junho de 2003.

REFERÊNCIAS BIBLIOGRÁFICAS

BARTIÉ, Alexandre. **Garantia da Qualidade de Software**. Rio de Janeiro: Editora Campus ltda, 2002.

BOGER, M.; STURN, T.; SCHILDHAUER, E. **Poseidon for UML Users Guide**. Disponível em: <http://www.gentleware.com>. Acesso em 17 de junho de 2003.

BOOCH, G. **Object-Oriented Analysis and Design with Applications** 2nd.Redwood City, CA: Ed. Benjamin/Cummings, 1994.

BOOCH, G. **Object solutions**. Redwood City, CA: Addison-Wesley, 1995.

BOOCH, G. **Object solutions: Managing the Object-Oriented Project**. Addison-Wesley, Reading – MA, 1996.

BOOCH, G. **Object Oriented Design**. Disponível em: <http://www.cs.ualberta.ca/~pfiguero/soo/method/ood.html>. Acesso em: 23 de abril 2003.

COAD, P. ; YOURDON, E. **Análise baseada em objetos**. Rio de Janeiro: Editora Campus, 1992.

COAD, P.; YOURDON, E. **Projeto baseado em objetos**. Rio de Janeiro: Editora Campus, 1993.

COMER, D. E. **Redes de computadores e Internet.** 2ª. ed. Porto Alegre: Bookman, 2001.

DEMARCO, T. **Structured Analisys and System Specification.** Prentice-Hall, 1979.

DEITEL, H. M. **Java, como programar.** trad. Carlos Arthur Lang Libôa. – 4.ed. – Porto Alegre: Bookman, 2003.

Documentações sobre a ferramenta Rational Rose. Disponível em: <http://www.rational.com>. Acesso em: 26 de março 2003.

DODD, A. Z. **O guia essencial para telecomunicações**. Tradução da 2ª. Ed. Original – Rio de Janeiro: Campus, 2000.

FILHO, W. P. P.; SANT´ANA, C. R. G. **Manual de Engenharia de Produtos de Software – Parte I: Recomendações**. RT – DCC – 008/1998a.

FILHO, W.P.P.; SANT´ANA, C. R. G. **Manual de Engenharia de Produtos de Software – Parte II: Padrões e Modelos**. RT – DCC – 009/1998b.

FILHO, W. P. P. ; SANT´ANA, C. R. G. **Manual de Engenharia de Processos de Software – Parte I: Políticas**. RT – DCC – 015/1998c.

FILHO, W. P. P. ; SANT´ANA, C. R. G. **Manual de Engenharia de Processos de Software – Parte II: Padrões e Modelos**. RT – DCC – 016/1998d.

FILHO, W. P. P. **Engenharia de Software: fundamentos, métodos e padrões**. Rio de Janeiro: LTC – livros técnicos e científicos editora, 2001.

FURLAN, J. D. **Modelagem de Objetos através da UML – the Unified Modeling Language**. São Paulo: Makron Books, 1998.

GANE, T.; SARSON, C. **Structured Systems Analisys.** McDonnell Douglas, 1982.

HATLEY, D. J.; PIRBHAI, I. A. **Strategies for Real-Time System Specification.** Dorset House, 1987.

HARMON, P. **Business Process Trends**. Disponível em: <http://www.popkin.com>. Acesso em: 17 de junho de 2003.

HUMPHREY, W. S. **A Discipline for Software Engineering**. Addison-Wesley, Reading – MA, 1995.

JACKSON, M.A. **Principles of Program Design**. Academic Press, 1975.

JACKSON, M.A. **System Development**. Prentice-Hall, 1983.

JACOBSON, I.; ERICSSON, M.; JACOBSON, A. **The Object Advantage: Business Process Reengineering with Object Technology**. Addison-Wesley, Reading – MA, 1994a.

JACOBSON, I. **Object-Oriented Software Engineering**. Addison-Wesley, Reading – MA, 1994b.

JACOBSON, I.; GRISS, M.; JONSSON, P. **Software Reuse: Architecture, Process and Organization for Business Success**. Addison-Wesley, Reading – MA, 1997.

JACOBSON, I.; RUMBAUGH, J.; BOOCH, G. **Unified Software Development Process**. Addison-Wesley, Reading – MA, 1999.

LARMAN, C. **Utilizando UML e padrões: uma introdução à análise e ao projeto orientados a objetos**. Trad. Luiz A. Meirelles Salgado. Porto Alegre: Bookman, 2000.

MELO, A. C. **Desenvolvendo aplicações com UML**. Rio de Janeiro: Brasport, 2002.

O´BRIEN, J. A. **Sistemas de informação e as decisões gerenciais na era da Internet**. São Paulo: Saraiva, 2002.

OPPENHEIMER, P. **Projeto de redes top-down**. Rio de Janeiro: Campus, 1999.

ORR, K. T. **Structured Systems Development.** Yourdon Press. Nova Iorque, 1977.

PAGE-JONES, M. **The Practical Guide to Structured Systems Design.** Yourdon Press, 1980.

PRESSMAN, R. S. **Engenharia de Software**. São Paulo: Makron Books, 1995.

QUATRANI, T. **Modelagem Visual com Rational Rose 2000 e UML**. Rio de Janeiro: Editora Ciência Moderna Ltda, 2001.

ROCHA, A.R.C.; MALDONADO, J.C.; WEBER, K.C. **Qualidade de Software – teoria e prática**. São Paulo: Prentice Hall, 2001.

RUMBAUGH J.; BLAHA M.; PREMERLANI W.; EDDY F.; LORENSEN W. **Modelagem e projetos baseados em objetos**. Rio de Janeiro: Editora Campus ltda, 1994.

RUMBAUGH, J.; BOOCH, G.; JACOBSON, I. **UML – guia do usuário**. Rio de Janeiro: Editora Campus ltda, 2000.

SHLAER, S.; MELLOR, S.J. **Object-oriented Systems analysis:Modeling the World in Data.** Englewood Cliffs, New Jersey, Yourdon Press, 1988.

SHLAER, S.; MELLOR, S.J. **Object Life Cycles: Modeling the World in States.** Englewood Cliffs, New Jersey, Yourdon Press, 1992.

SOARES, L.; LEMOS, G.; COLCHER, S. **Redes de Computadores: das LANs, MANs e WANs às redes ATM.** Rio de Janeiro: Editora Campus ltda, 1995.

STEVENS, W. P.; MYERS, G.P.; CONSTANTINE, L.L. **Strutured Design**. IBM Systems Journal, vol.13, n.2, 1974.

TANENBAUM, A.S. **Computer Network.** 3a. ed. Englewood Cliffs, Nova Jersey: Prentice Hall, Inc., 1996. Publicado no Brasil pela Editora Campus com o título *Redes de computadores*.

WARD, P. T.; MELLOR, S. J. **Structured Development for Real-Time Systems.** Yourdon Press, 1985.

WARNIER, J. D. **Logical Construction of Programs.** Van Nostrand Reinhold, 1974.

YOURDON, E. N.; CONSTANTINE, L. L. **Strutured Design.** Yourdon Press, 1978.

YOURDON, E. N. **Análise estruturada moderna**. Rio de Janeiro: Editora Campus ltda, 1992.

ANEXO A

Figura 60 – Diagrama de atividade: ciclo do sangue